FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Mina Ghobrial

Fish Detection Automation from ARIS and DIDSON SONAR Data

Master's Thesis
Degree Programme in Computer Science and Engineering
June 2019

Ghobrial M. (2019) Fish Detection Automation from ARIS and DIDSON SONAR Data. University of Oulu, Degree Programme in Computer Science and Engineering, 60 p.

ABSTRACT

The goal of this thesis is to analyse SONAR files produced by ARIS and DIDSON manufactured by Sound Metrics Co. which are ultrasonic, monostatic and multibeam echo-sounders. They are used to capture the behaviour of Atlantic salmon, which recently has been on the lists of endangered species. These SONARs can work in dark lighting conditions and provide high resolution images due to their high frequencies that ranges from 1.1 MHz to 1.8 MHz. The thesis goes through extracting data from file, redrawing it, and visualising it in human friendly format. Next, images are analysed to search for fish. Results of analysis are saved in formats such as JSON, to allow harmony with other legacy systems. Also the output helps in future development due to the support for JSON in multitude of programming languages. Eventually, a user-friendly user interface is introduced, which helps making the process easier. The software is tested against data-sets from rivers in Finland, that are rich in Atlantic salmon.

Keywords: Machine Vision, Morphological Operations, Underwater Acoustics, Hydroacoustics, Ultrasound, Ultrasonic Imaging, Dual-Frequency Identification SONAR, Fisheries.

TABLE OF CONTENTS

ABSTRACT
TABLE OF CONTENTS
FOREWORD
LIST OF ABBREVIATIONS AND SYMBOLS
1. INTRODUCTION 7
1.1. Background 7
1.2. Overview 8
1.3. Objectives and Challenges 9
1.4. Contribution 9
1.5. Structure 10
2. SONAR 11
2.1. Hardware Description 12
2.1.1. Lenses 13
2.1.2. Transducer Array 14
2.1.3. Beam Formation 14
2.1.4. Image Formation 17
2.2. Setup and Software 18
2.2.1. Setup 18
2.2.2. Software 18
2.2.3. Data Format 19
3. COMPUTER VISION 23
3.1. Background Subtraction 23
3.2. Morphological Operations 25
3.3. Object Labelling 26
3.4. Automation Examples 27
4. IMPLEMENTATION 30
4.1. File Operations 30
4.1.1. Read 30
4.1.2. Visualising Frames 31
4.1.3. Write 32
4.2. Proposed Framework 33
4.2.1. Preprocessing 33
4.2.2. Background Subtraction 33
4.2.3. Morphological Operations 34
4.2.4. Connected Object Labelling 35
4.2.5. Tracking 35
4.2.6. Output and Manual Override 37
4.3. User Interface 38
4.4. Work-flow 39
4.5. Output 41
5. EXPERIMENTS & DISCUSSION 43
5.1. Tuning of Variables 43
5.2. Analysing Batches 44

5.3. Exporting Results 45
5.4. Future Work 45
6. CONCLUSION 47
7. REFERENCES 48
Appendices 51
A. ECHO-SOUNDERS HARDWARE SPECIFICATIONS 52
B. FILE AND FRAME HEADERS 54
C. OUTPUT TEMPLATES 59

FOREWORD

This thesis work is mostly concerned about preserving the natural resources. Which should help maintaining one of the endangered species, namely salmons. I have grown interested more and more as I went through the details of this topic, which I will try to define all the aspects of in the following chapters of this thesis. I wish to mention some people who had the most influence on my life, my goals, and my achievements. All the words in the world can't describe how grateful I am to my family, who supported me since my childhood towards my dreams. I would like also to a huge thanks to my friends and my colleagues for helping and supporting whenever I needed them. Most of all, I wish to mention my fiancee for her great support all the time, and being my constant motivation. Last but not least, an enormous thanks to my professors who provided me with all the knowledge that I now posses.

Oulu, 17th June, 2019

Mina Ghobrial

LIST OF ABBREVIATIONS AND SYMBOLS

DIDSON	Dual-Frequency Identification SONAR
ARIS	Adaptive Resolution Imaging SONAR
SONAR	Sound Navigation Ranging
SARA	Species At Risk Act
ESA	Endangered Species Act
DFO	Department of Fisheries and Oceans
LUKE	Luonnonvarakeskus, The natural resources institute in Finland
FOV	Field of View
RR	Range Resolution
LR	Long Range
IF	Identification Frequency
DF	Detection Frequency
CSE	Computer Science and Engineering
GUI	Graphical User Interface
MoG	Mixture of Gaussian
f	Frequency
λ	Wavelength
v	Sound Velocity
d	Total distance covered by sound
t	Time sound needs to cover a distance d
r	Distance between an object and SONAR
N	Number of Beams emitted by a SONAR or number of active transducer array units
M	Number of samples per beam
p	Number of ping cycles per frame produced by a SONAR
$\mathbf{R}$	Orthogonal matrix
r_{ij}	Matrix element
$\mathbf{v}$	Velocity vector
$\mathbf{x}$	Planar coordinate

1. INTRODUCTION

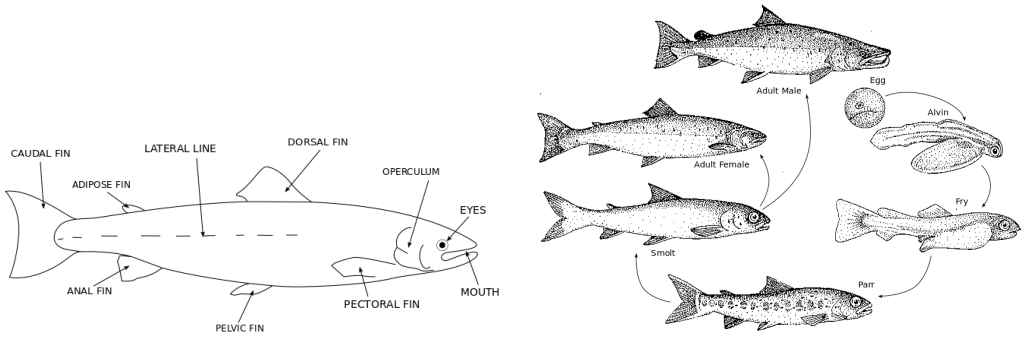
1.1. Background

Salmons belong to a family of ray-finned fish which is called Salmonidae. This family has a genus called Salmo, which Atlantic Salmons belong to. The binomial name of Atlantic Salmon is Salmo Salar. Salmonidae are mostly characterised by their Pelvic fins placed on the far back, and an Adipose fin towards the rear of the back. They are also characterised by their length with maximum of 2 meters. However, Atlantic Salmon is the largest species in its genus, where the average weight is around 3.6 to 5.4 kg, while it reaches 13 kg sometimes. The species has an average length of about 75 cm [1].

The species' life span ranges from 3 to 7 years, and it's an anadromous species, which means that it can live in both salty and fresh water. Its life cycle, as shown in Figure 2(b), is complex, first it is born in fresh waters as an Alvin, then it migrates to salty waters as Smolt [2]. it spends long portion of its life in salty water feeding, and growing until they become mature enough to spawn. Eventually, they migrate again to fresh water like juvenile rearing in rivers to spawn, during Summer or Spring seasons. The water is usually 0.5 to 3 meters deep. The process takes up to two weeks. Most Salmons die after spawning, with very low probability of survival and re-spawning in later seasons, with diminishing probabilities to nearly none for spawning for a third time. The area where the species lives is highlighted in the map in Figure 1.

Figure 1. Atlantic Salmon distribution map

In Finland, only Atlantic Salmon can be found, which is listed in the Endangered Species Act (ESA) alongside other endangered Salmon species [3], but the only type listed under Canada's species at risk act (SARA) [4]. Its stocks had been decreasing throughout the years, since the 1970s, due to different life conditions in each of the ecosystems, the species is threatened by multiple types of threats. One type of threats is a result of the human impact on the environment, for example, fishing with nets [5] in the old days and recently recreational and commercial fishing, also building impediments across the river streams like dams



(a) (b)

Figure 2. (a) External Atlantic Salmon anatomy. (b) Atlantic Salmon life cycle.

without fish passages [6], and modifications in the spawning and rearing habitat [7], water quality deterioration.

In order to preserve such species, quite a lot of considerations have been taken, and a lot of organisations, entities, and even individuals have taken some strict measures to help the situation. Like the final Recovery plan for the Gulf of Maine Distinct Population Segment of Atlantic Salmon provided by the ESA [8].

1.2. Overview

The Natural Resources Institute of Finland (Luonnonvarakeskus, also known as LUKE) has been working on the same issue of preserving such species, by monitoring the number of Salmon going upstream in two of the most important rivers in Finland, Tornionjoki and Simojoki rivers. On a yearly basis, in the period between end of May till the end of August, is the spawning season of Salmon. Echo-sounders are placed underwater in both rivers, where they produce a stream of sonograms. Such stream of sonograms is utilised to detect adult Salmons to a distance of about a hundred meters from the echo-sounder unit. The complication with echo-sounders is that they don't provide a highly detailed sonogram when used to cover such long distances, like the Salmon fish indicated in Figure 3. Hence, usually personnel who monitors the captured stream of sonograms rely on their prior knowledge about Salmon physical characteristics (i.e. length) and their experience to decide whether what appears on their screen is a Salmon or not. The results of such process are published publicly on their website [9].

Salmon has distinguishable visual characteristics, indicated in Figure 2(a) such as a tapering, streamlined body, and an anal fin which has maximum of 12 rays. The one characteristic that makes it stand out among other species, is that the wrist part to the tail is narrower than any other species. However, this level of detail is only accessible whenever the echo-sounders are only set to capture short ranges [10].

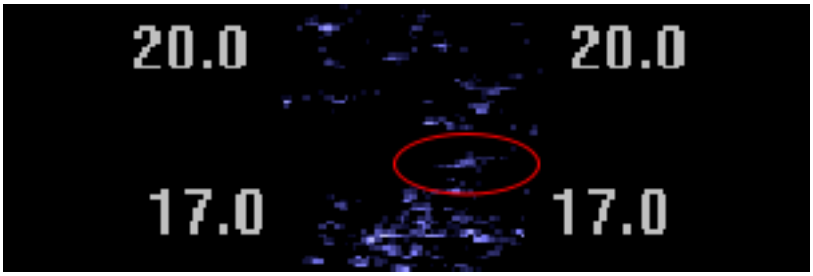


Figure 3. A sonogram showing a detected fish at around 18m from the echo-sounder.

1.3. Objectives and Challenges

The process of manually counting the Salmon crossing the echo-sounders is challenging and laborious. A team of observers has to dedicate nearly their whole Summer and Spring seasons of the year to the task, where twice a day they reach to the echo-sounders to acquire the collected data from the on-site storage units. They analyse 24-hours worth of data daily, by observing the captured stream of sonograms. Whenever a Salmon is spotted in the sonogram, its length is measured manually, then added to the stack of detected fish. Each sonogram stream is analysed at least twice and the results are compared to settle on the number of the detected number of Salmon. Results are exported into a text file with some pre-formatted template.

The process is tedious and highly error prone and time consuming. The most considerable drawback is the quality of the captured sonogram, which is usually extremely low and can not be reliably classified by a beginner, and usually takes some time learning and adapting the patterns of movement of Salmon. Sometimes a one-meter long Salmon might appear on the screen as couple of pixels.

The objective of this work is to present an automated system to analyse the captured stream of sonograms, detect the Salmons, calculate their sizes, detect their movement direction and other parameters, and finally export the result into various more useful formats other than text, to allow the integration of the system with multitude of other platforms and frameworks, like web-pages. This work proposes a framework for analysing the sonograms alongside a computer program with a properly designed user interface to ease the sonogram analysis.

1.4. Contribution

This work uses machine vision methods in analysing the sonogram, by subtracting the background then performing morphological operations on the resulting visuals, and finally detecting objects and track them through their screen life-time. In addition, it produces the results in multiple formats, supporting legacy systems and providing the opportunity for integration with various other platforms.

1.5. Structure

The thesis is structured as follows: first, Chapter 2 provides a literature review about the history of SONAR devices, their recent data format output, and pieces of software used with such hardware units. Next, Chapter 3 presents a review of machine vision tools that will be used in the implementation chapter, alongside other relevant tools, that might as well be used. Chapter 4 presents the proposed framework and its implementation. Then, in Chapter 5, experiments and their results are presented. Eventually, a comparison is formed to show the differences in performance between the manual and the automatic systems, alongside the conclusion.

2. SONAR

The first thing to think of, when trying to solve a problem, is understanding it correctly from all aspects. To be able to protect such an endangered species, we need to study its behaviour. Aquatic creatures are the hardest to study, because most of the processes to study such organisms are an invasive processes, which affect the ambient environment or the life style of the organism under study. The most non-invasive way to examine such creatures is to observe from distance by capturing images that show their behaviour.

Due to the lack of good lighting under the surface of rivers' fresh water, a camera is not useful for capturing any visual information. With the water being turbid or dark and usually not clean, light is never reflected nor caught by the camera. This is when an acoustic camera [11] [12], a SONAR or an echo-sounder comes in handy. It has relatively high resolution and refresh rate.

A SONAR is an acronym for Sound Navigation Radar, which is device that uses a mechanical wave (sound in our case) reflections to detect and identify surroundings. There are two types of SONAR, Passive SONARs which only listen to waves produced and/or reflected by other bodies, and Active SONARs which beams out wave pulses (either unidirectional or omnidirectional) and listens to their echoes. The technique is mostly used for detecting the direction and the distance of the surrounding environment from the source. The first echo-sounder was first devised by the German scientist Alexander Behm [13] after the Titanic disaster in 1912, to be used by ships in detecting icebergs and avoiding them. The technology was later used in World War I. While the use of the word SONAR was evoked in World War II, by the Americans.

The active SONARs are categorised into categories based on the location of the transmitter and the receiver. Monostatic [14], means that the transmitter and the receiver are placed together. While Bistatic, the transmitter and receiver are separated by a distance that correlates the expected distance from the target. On the other hand, a Multistatic SONAR is having more transmitters and/or receivers. Also they can be categorised based on the number of beams that they transmit, into single-beam and multi-beam systems, like shown in Figure 4. Thus, using multi-beamed systems produces higher resolution output.

Such technique is extremely useful when trying to capture information under the surface of dark areas in water or may be turbid water, which is considered non-invasive method as it does not need any tactile means to gather information which was mainly used before this invention. It has multitude of applications [15] in different fields, for instance, *Identification* & *Inspection* of objects and/or faulty hardware under waters, *Guidance* for underwater vehicles, *Monitoring* the behaviour of aquatic creatures [16] [17] and *Detection* of intruders. The latter was the main reason for developing such technology.

The current active multi-beam echo-sounders used in most places, belong to Sound Metrics Co. [18] which produces two main families of echo-sounders namely Dual-Frequency Identification SONAR (**DIDSON**) and Adaptive Resolution Imaging SONAR (**ARIS**). Henceforth, all the information and/or text will be explaining DIDSON, unless otherwise stated. Because ARIS is basically an

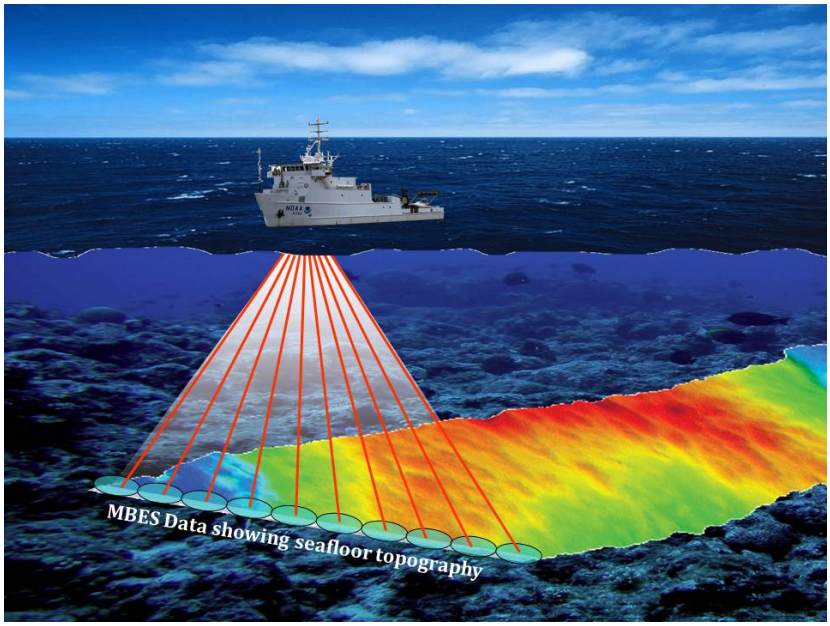


Figure 4. An image showing a ship equipped with a multibeam SONAR for mapping the topography of seabed

evolution from DIDSON, which is designed for higher resolution and capturing more information.

2.1. Hardware Description

The mentioned SONARs belong to a category of active SONARs called Monostatic SONARs, which has both the transmitter and the receiver at the same place, like shown in Figure 5. They are also in the multibeam systems category. All versions of hardware and their corresponding specifications are mention in A. It generates a ping (a pulse), and measures the time t for the ping to hit the target and gets reflected and captured by the receiver. Knowing the speed of sound v in the medium at hand, we can easily calculate the distance r between the echo-sounder and the target using $v = f\lambda$, $d = 2r = vt$.

Both **ARIS** and **DIDSON** have two types of frequencies, Identification Frequency (**IF**) and Detection Frequency (**DF**). Both modes operate in the Medium Frequency Range (**MF**) as Radio waves, with the Identification frequency being the higher, and Detection frequency the lower. Identification mode covers shorter distances, using higher frequencies and more transmit/receive cycles. Detection mode covers longer distances, using lower frequencies, with less transmit/receive cycles. However, in both modes the device maintains the same frame rate and power consumption. The device consists of 4 main pieces that are essential for its operation, acoustic lenses, transducer array, focus motor, and the mechanism to move the secondary lenses.

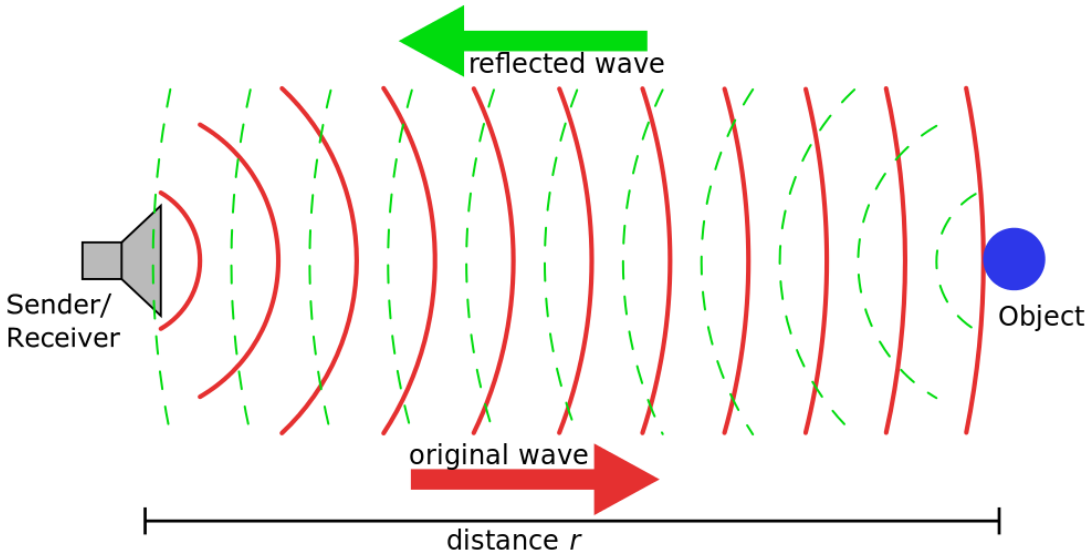


Figure 5. Diagram of the process of sound emission by a monostatic SONAR and reflection from an object.

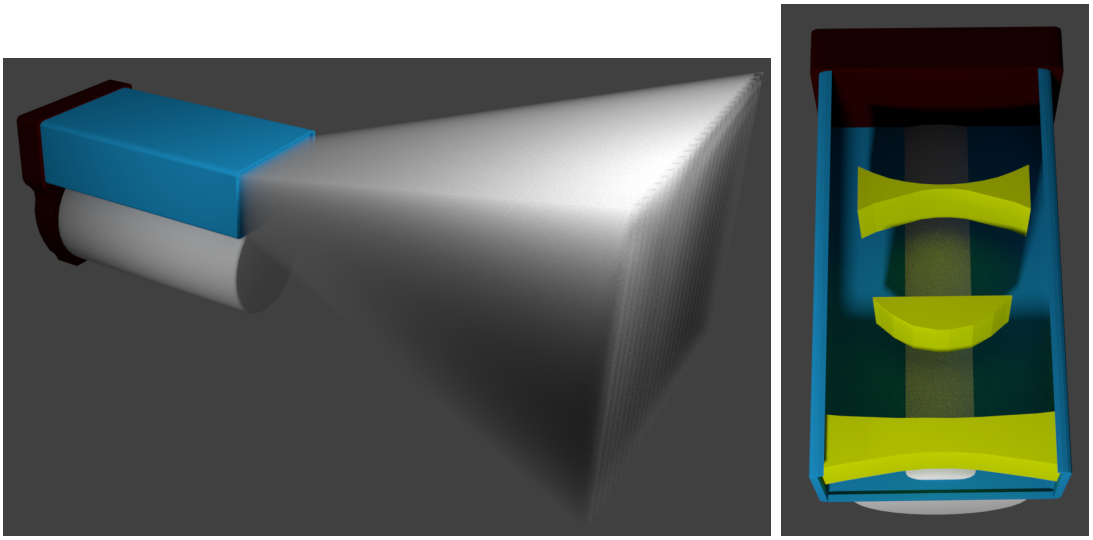
2.1.1. Lenses

The most important component of the whole device is the acoustic lens, which is used to direct and optimise the focus of the device's generated waves through the whole detectable range. They are also responsible for transmission and reception of narrow beams [19].

Like optical lenses, acoustic lenses are used to control directivity (convergence/divergence) of sound waves, as it is made of a material which has a non-unity refractive index n which has different sound speed than the ambient medium. This is useful in many applications in multitude of fields, including this work's. The material for making such lenses is called polymethylpentene, because it has an impedance which is close to water's. Hence, efficiency is maximised as only less than 1% of the incident waves are reflected back.

The idea of acoustic lenses has its advantages, as using such lenses consumes no power. However, it also has its own disadvantages, because they take up a lot of space in front of the transducer array, making the device bulky, and also the countless reflections produce reverberation. However, such reflections have very low power and they do not produce ghosting noise, but they just contribute slightly to a brighter noisy background. From Figure 7, L1 is composed of 3 materials, biconcave plastic lens, fluid, and thin plastic lens, from left to right respectively. L2 is a Convex-planar plastic lens, moved by the mechanism, changing distance between L1 and itself to change focus. L3 is a concave lens used to direct acoustic waves from the transducer array T, which has N units that transmit/receive a beam, thus it corresponds to the number of beams produced by the device model.

The design of the acoustic lenses is an iterative process, where computer simulations are used to produce multiple designs and analyse their wave



(a) (b)

Figure 6. Diagram for one of the DIDSON SONAR devices version. (a) Diagram shows the 3D structure of the device and its Field of View. (b) Diagram shows the Acoustic lenses positioning inside the SONAR device.

propagation efficiency. Then the designs are compared together to select the best combination of beam-width, sidelobes, lens loss for each beam, and how ambient water conditions from salinity and/temperature will affect the operation of the device. After choosing the best design, lenses are fabricated.

2.1.2. Transducer Array

The linear array of transducers T, shown in Figure 7, has number of transducer elements which is equal to the maximum number of beams that the device can produce. The array is made of a composite material called PZT 1-3, which provides a wide bandwidth for the device operation, with IF and DF as the cut-off frequencies of a pass band filter.

When the device operates at the highest possible frequency (IF) all the elements of the transducer array are working (transmitting/receiving), in contrast with working with the lowest possible frequency where only half of the transducers are working. Regardless of the working frequency, the device can either operate on full load, where all transducer units are working, or on half load, where only the even half of those units are working.

2.1.3. Beam Formation

The acoustic lenses control the horizontal beam-width, while the curved transducer array controls the vertical beam-width. When a focused line of sound waves coincides with the long thin transducer element, acoustic energy is

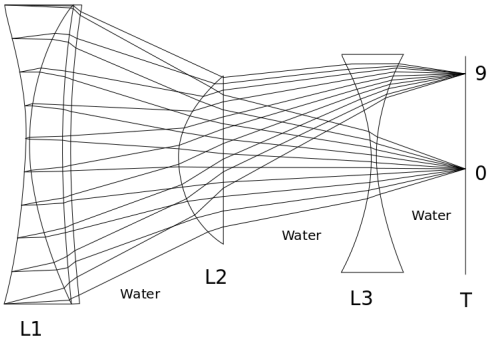


Figure 7. Diagram showing lenses inside one of the DIDSON devices collecting 2 plan waves' ray-traces, at 0°, perpendicular to the first lens, and 9° degrees off the perpendicular. (Diagram is just an illustration and is not to scale)

transformed into electrical energy so that it can be processed later. The SONAR loses sensitivity when the beams received are more skewed off-axis. Thus, sound waves are only transmitted and received in specific directions while any other different direction produces some type of imaging noise.

To avoid such noise, this SONAR type uses ping cycles [15], where only q transducer units, which are distributed evenly through the whole transducer array, transmit or receive at the same time. Each frame captured by the SONAR is formed using p number of ping cycles, which depends on the number of active transducer units using equation $p = N/q$. The number q depends on the horizontal FOV and the produced beam-width.

The produced beam-width is often defined differently across the industry, but EdgeTech [20] clarified the concept. Beam-width is the angle at which the beam's energy has reached **70%** of its peak, or the equivalent **-3dB** in decibels as shown in Figure 8. This shows the intensity of the radiated energy as a function of angle off the main lobe of the transducer. This is called the one-way beam-width which is either transmit only or receive only.

There are two types of beam-widths depending on the type or the mode of operation of a SONAR device, first is one-way beam-width which is the beam-width for a beam in one journey, either transmitted or received, second is two-way beam-width which is the beam-width for a beam from a SONAR operating in both transmitting and receiving modes.

In this case it is two-way beam-width because the SONAR is both transmitting and receiving through the same transducer array. In this mode, the beam-width is narrower and the net result two-way beam-width is the squaring of the one-way beam shape. It can be approximated using the following equation $\theta_2way = \theta_1way * 0{,}72$. One of the inevitable downsides of this setting, is the beams that fall skewed on the transducer array units. It produces sidelobes that can potentially ruin the image resolution if not designed carefully.

Figure 9 shows the detected beams from 3 array units receiving at the same time, with the centre beam at 0 degrees. The mainlobe is centred at the unit,

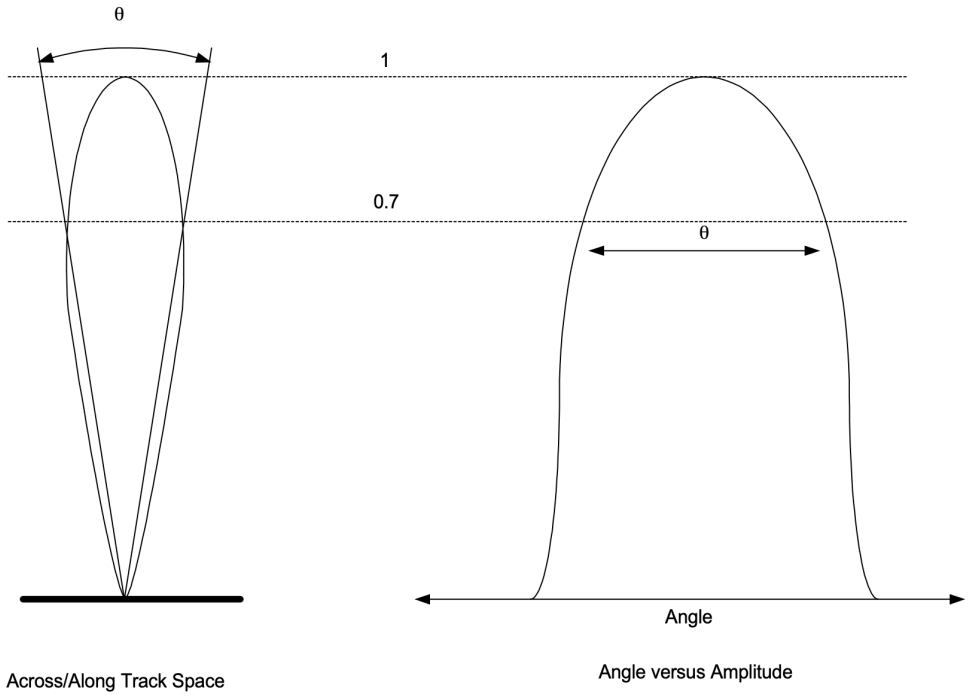


Figure 8. On the left, the directivity graph of one unit of the transducer array. On the right a graph showing the angle vs. the amplitude, while sidelobes are just hidden for the sake of clarification.

and the sidelobes extend in both left and right directions. In the one-way mode, the higher beam-width provides more powerful sidelobes which can be around **-13dB** lower than the mainlobe, which can affect the produced image resolution. However, in the two-way mode, the narrower beam-width, lower power sidelobes are produced, which are **-26dB** or **5%** from the mainlobe level. Thus, it produces better, sharper and higher resolution images. Also from the same figure, we can detect the proper spacing between each unit or the number of units q working simultaneously, that can produce the lowest sidelobe interference and reduce ghosting effects in the produced image.

Example: with a DIDSON SONAR version 300m, shown in Figure 6, it has 96 transducer array units and 29 degrees horizontal FOV (check Appendix A). It has two modes: one is Identification, which uses the whole 96 units and operating frequency around 1.8 MHz, or Detection which only uses 48 units, and operating frequency around 1.1 MHz.

Determining the number of beams q that can be produced in the same ping cycle is done using Figure 9, where the separation between peaks has to be minimised but at the same time avoids interference. So it has to be about 3 degrees (1.5 for each side of the mainlobe). Then dividing the horizontal field of view by 3, and ceiling the result to the nearest divisor of the total number of beams (96), we get 12 units. Thus, $\boldsymbol{q = 12}$, is the number of active units of that model per

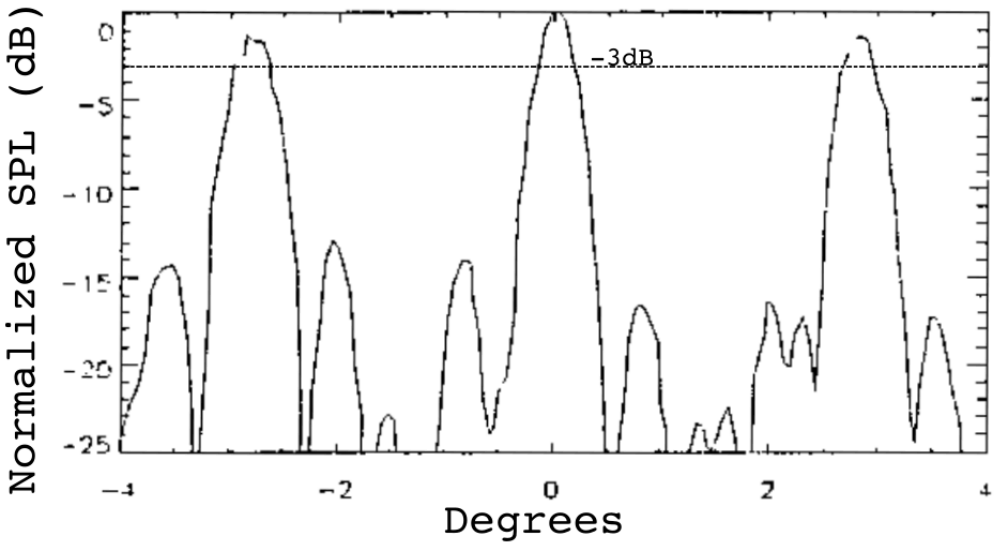


Figure 9. Showing the Normalised Spectral Power Level vs. Degrees of beam spacing, of 3 centre units output, from a set of 12 units from a ping cycle. With the centre transducer unit beam. Produced from DIDSON 300m.

ping cycle. Hence, no ghost images for any targets is formed in the final image. These numbers are calculated during the lens manufacturing processes.

- **Identification Mode** The number of ping cycles p can be calculated with N being 96, then p=8 ping cycles.
- **Detection Mode** The number of ping cycles p can be calculated with N being 48, then p=4 ping cycles.

2.1.4. Image Formation

As discussed in the previous subsection, each unit in the transducer array transmits a pulse and receives the resulting echoes which are confined in the narrow horizontal beam-width. The amplitudes, which maps to the distances of the objects from the SONAR, are transformed into pixel values and pixel locations on the produced final image. The number of beams of the used hardware decides on the horizontal dimension of the raw image, such that each beam is a column of pixels. So, a device with 96 beams has 96 columns of raw data pixels. However, the number of pixels in each column depends on the operating parameters of the device hardware. A device can be operating on Identification mode, so it produces the highest quality images it can, since one pixel maps to the smallest value of the range resolution, while on Detection mode, produces the lowest quality. However, the raw format of the data is not useful for further analysis nor processing, because it is laid down in Cartesian coordinates system, disregarding beam-widths, FOV, and basic wave properties. In order to start working with the captured data, it has

to be transformed into human-eye friendly format, which is the Polar coordinates system. The Cartesian coordinates frame layout is shown in Figure 18(a) and the Polar is shown in Figure 18(b). Henceforth, whenever the word *image* is mentioned, it means the human-eye friendly Polar coordinates image. However, whenever the word *frame* is used, it means the actual raw data in Cartesian coordinates that was captured originally by the SONAR.

2.2. Setup and Software

Understanding the SONAR device characteristics enabled researchers to choose the best setup and placement to help getting the best results. Hence, the setup subsection discusses the placement of the device in nature and how it is used by LUKE. Also the means by which data is recorded, analysed and its format inside a SONAR data file.

2.2.1. Setup

The DIDSON device is placed underwater in the area of the counting process, as the example show in Figure 10. Usually, steel mounts are used for placing the device in the intended locations. The device has to be fully immersed under water surface, allowing the voids to fill the insides of the device with ambient water to eliminate the air gap between the lenses inside the device [19]. The setup is surrounded by deflection weirs to prevent fish from swimming too near the SONAR unit. The vertical levelling of the unit is often changed on-site according to the water level, while the pan and tilt can be controlled remotely. The ideal FOV, shown in Figure 11 is when the beams uppermost samples are parallel to the water surface, while the lower samples span the whole river bed along the window length. The device is often set to create new files at every specific interval of time, which is usually one hour [21]. Thus, output files have time and date stamped names. Also the unit has to be lightning-proofed by grounding it properly to avoid frying cables and/or devices.

2.2.2. Software

The unit is connected to a computer on-site, which runs the latest software from Sound Metrics, as shown in Figure 12. The software is set to capture the stream of information coming from the unit through Ethernet cables. Captured information is saved to an external storage device with large volume. Hence, the responsible party can extract data from the software by just unmounting the external storage device and plugging another, for continuing the recording operation.

The data collected is analysed afterwards, by a dedicated team, in the following manner:

Figure 10. A DIDSON unit placed in Tornio river, in Kattilakoski, on the borders between Finland and Sweden.

- Enabling background subtraction, and set the playback speed to about 10 times the capturing speed.
- If the observer spots a fish on the screen, the playback is stopped, and the image is magnified.
- The observer plays the set of frames at which the fish was visible, until the clearest image is settled upon.
- Fish length is measured by using a marker tool to mark the beginning and the ending tips of the fish on the screen. Then it is entered into the output file.

This method for analysing data is highly error prone and time consuming. Hence, the need for an automatic software rises.

2.2.3. Data Format

There are two families of devices indicated in Appendix A. Thus, we have two main file formats, which are `.aris` for ARIS and `.ddf` for DIDSON. However, `.ddf` has been developed multiple times over the years, so it has many versions, ranging all the way from `ddf_00` to `ddf_04`. Also, `.aris` is considered `ddf_05`. Mainly all of them has the same basic format, which is, every file starts with `File Header` which specifies how the data is laid down in the file. Then it has i number of frames, that ranges from 0 to *i-1*, where each frame has its own `Frame Header` which specifies the properties of the captured frame.

The differences between each format is the layout of data inside. Table 1, is a comparison between all the file versions, according to their file and frame

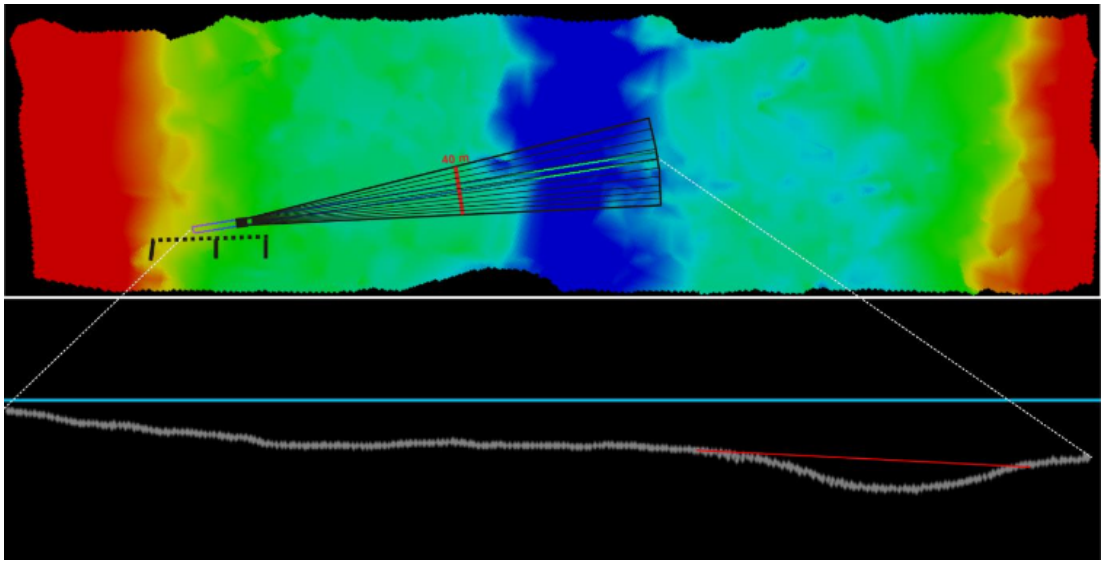


Figure 11. Diagram showing the range that the unit mentioned in Figure 10. The device was used in this setting to cover a range of 80m in total. It also shows the ideal FOV, where the produced beams nearly cover the whole river bed. However, the area under the red line is not covered.

Table 1. SONAR file format versions

	ddf_00	ddf_01	ddf_02	ddf_03	ddf_04	ddf_05 (ARIS)
Version Number	0x00464444	0x01464444	0x02464444	0x03464444	0x04464444	0x05464444
File Header Size (Bytes)	variable	variable	512	512	1024	1024
Frame Header Size (Bytes)	120	120	256	256	1024	1024

headers' sizes, alongside their version numbers. However, the general file format is displayed in Figure 13. The main differences between each file format is the number of bytes that represents each of the headers and data size. Also newer hardware versions have more modifications and add-ons, for example, an attached GPS and motion motor for moving the device, where the values of each are also added to the headers.

The first 3 versions of the format are in some way obsolete, because they are relatively old and they are not in use anymore. Henceforth, only the latest 3 file formats will be considered.

The files are laid down in binary format, so it is indexed in bytes as units. Each group of bytes represent one property or one configuration of the data inside the file. The first 4 bytes always contain the version number. It can be seen from Table 1 that the version number of each file is represented by 8 hexadecimal characters, which means it is contained in 4 bytes of the file. It is often used by any piece of software to indicate the version of the file and if the file is corrupted or not.

Figure 12. An image showing a laptop placed in a protective lightning-proof container near Tornio river in Kattilakoski site, Finland.

The file arrangement looks like shown in Table 2. It always starts with the file header, which starts at byte zero, and occupies a number of bytes which is equal to FILE_HEADER_SIZE, which is a constant number of bytes in each file depending on its format. Next, comes the whole first frame (frame index = 0), which starts with the header of the first frame at byte number FILE_HEADER_SIZE, and continues to byte number (FILE_HEADER_SIZE + FRAME_HEADER_SIZE) where FRAME_HEADER_SIZE is a constant size through the whole file depending on its format. Then afterwards comes the captured data for the first frame (frame index = 0). The size of the captured data depends on the number of active beams and the number of samples captured across the range of each beam, which mainly maps to the mode of operation of

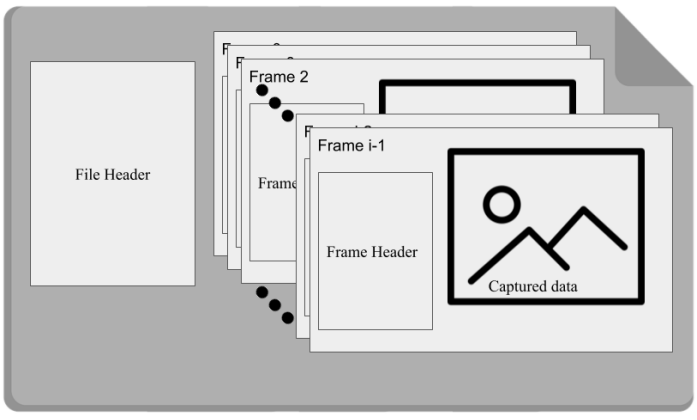


Figure 13. Diagram showing the general format of a SONAR file.

Table 2. The layout of data in any SONAR file format.

Start Location	Number of Bytes	Contents	Description
0	FILE_HEADER_SIZE	File Header	Each file has only one file header, which starts at byte 0.
0 + FILE_HEADER_SIZE	FRAME_HEADER_SIZE	Frame Header	Header of frame number 0.
FILE_HEADER_SIZE + FRAME_HEADER_SIZE	Captured Data Size: Beams * Samples	Frame data	Contains frame 0 captured samples data from each beam.
FILE_HEADER_SIZE + Whole frame 0 data	FRAME_HEADER_SIZE	Frame Header	Header of frame number 1.
FILE_HEADER_SIZE +Whole frame 0 data +FRAME_HEADER_SIZE	Captured Data Size: Beams * Samples	Frame data	Contains frame 1 captured samples data from each beam.
...	...	...	...

the SONAR device. The rest of the file continues with the same sequence, file header, frame 0 header, frame 0 data, frame 1 header, frame 1 data, and so on, until it reaches the end of the file (frame index = *i-1*).

Example: In the ARIS file format, FILE_HEADER_SIZE is equal to 1024 bytes and the FRAME_HEADER_SIZE is also 1024 bytes. Thus, the file header starts at byte zero and ends at byte 1023. Next, the first frame header starts at byte number 1024 and ends at 2047, then the first frame data starts at 2048 and occupies a number of bytes equal the number of active SONAR beams multiplied by the number of samples captured per beam, where these values are file-specific, so they are mentioned in the headers inside each file.

3. COMPUTER VISION

In the field of machine vision, images are usually transformed from representations that are easily visualised by humans, to simpler forms that are easily understood by a machine. Thus, after capturing data from underwater SONARs, it's essential to find the right methods to visualise these pieces of data, then transform them into machine-friendly formats for further analysis.

Through this chapter, multiple image processing techniques and algorithms are presented, to help representing and analysing data at hand, which will serve as basis for choosing specific algorithms in the implementation, in Chapter 4.

The last section of this chapter shows automation trials from multiple groups in separated parts in the world, which proves the potential of such field in future research.

3.1. Background Subtraction

Background subtraction is a very popular technique used to identify objects in motion in a sequence of captured frames. For example, the problem of detecting moving cars in an urban traffic video footage. The literature is full of methods for approaching such problem. Differences between each, are the ability of these methods in handling different luminance conditions, weather conditions, cast shadows, and the cases when moving objects go into rest [22].

The technique is used in various fields, like traffic monitoring in detecting vehicles violating traffic rules, human detection and tracking in surveillance systems, and gesture recognition.

Despite the numerous approaches, all follow a generic model shown in Figure 14, which consists of four main processes: Preprocessing, Background Modelling, Foreground Detection, and Data Validation.

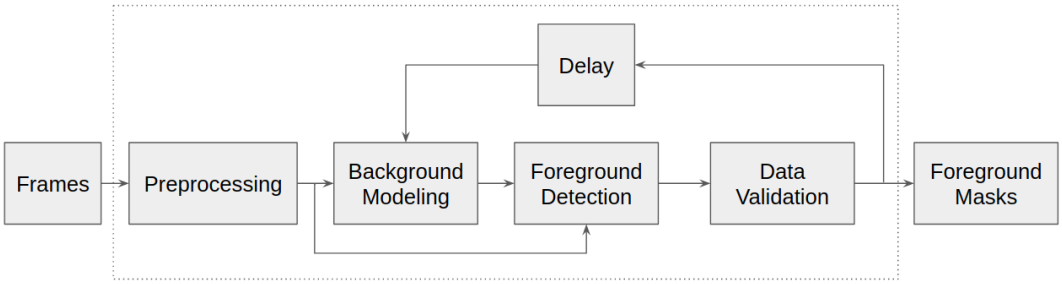


Figure 14. Generic flow diagram of a background subtraction algorithm.

Preprocessing is a set of simple operations performed onto input frames, that prepares the data for processing stage by increasing data quality. For instance, one of the operations might be temporal or spatial smoothing the input frames to remove camera noise from the process of recording to remove unwanted environmental conditions like rain or snow. Also sometimes the preprocessing stage is used to perform some little modifications to the input frames, like reducing

the frame rate. This enhances the speed of the algorithm. It might also be used to perform colour space changes from RGB or HSV spaces to more simpler spaces like greyscale. Thus, reducing the computational complexity of the algorithm, because most of which keep an independent model for each pixel, since having a RGB or HSV representations, adds more channels.

Background modelling is the heart of the algorithm, which has to be designed carefully, to be robust against undesirable artefacts, suitably adaptive, not resource-hungry, and at the same time capable of determining all the needed moving items. The main disadvantages of most background modelling schemes are, that they ignore neighbouring pixels correlation, the adaption rate may not correspond with the motion to be detected, and the detection of undesirable objects' motion from the background.

The algorithms are classified according to their recursiveness to recursive and non-recursive algorithm:

- Non-recursive models, store L frames in a buffer, and use temporal sliding-window approach to estimate the background image through the variation of each pixel within buffer frames. If L is large, the model takes more memory. This is used when detecting slow motion. However, the needed number of frames decreases if the motion to detect is relatively faster. These models are highly adaptive as they depend only on a subset of the past frames. Examples of such models are, frame differencing, median filtering, and linear predictive filtering.

- Recursive models, do not use buffers, which means much lower memory usage. However, they use periodic updating schemes to keep one single background model in memory. Thus, all input frames can have a varying effect on the current model at any point in time, which is considered a mixed blessing. In other words, if an error happens in the background model, it will stay for longer time than non-recursive models. But such disadvantages have been dealt with differently throughout the literature. Examples for such models are, approximated median filtering, Kalman filtering, and mixture of Gaussian (MoG), which is used in this work, and will be mentioned later in the implementation in Chapter 4.

Foreground detection is the process which uses the resulting background model to analyse new input frames and determines candidate foreground pixels, by checking if each input pixel is significantly different from the background model or not.

Data validation is the process of dealing with the disadvantages of the background modelling algorithms previously mentioned. For instance, to solve the problem of correlation between neighbouring pixels, a common way is to use morphological operations and connected object labelling to avoid any discontinuity in any of the detected objects.

The problem of incompatible adaption rate, produces false foreground masks which is also known as ghosting. It can be solved by using multiple background models with different adaption rates for each, and then cross-validate between the output of each model.

However, the problem of detecting undesirable objects or shadows is still troublesome. But most of the approaches use MoG combined with morphological operations.

3.2. Morphological Operations

The field of mathematical morphology is used to give quantitative description of shapes in an image. This field is based upon logical relations between pixels. Morphological operations are a set of image processing techniques that consists a set of morphological operators that can change the form of an input image according to a set of parameters. The two elementary operators are erosion, dilation. The two elementary operations can be combined in a specific sequence to produce other operations like opening, and closing. The purpose for such complementary operations is to remove unwanted details in images whose dimensions are smaller than the kernel used, while preserving the global shapes. All the previous operators were developed to operate on binary images. However, later generalisations have been made. The next definitions are based on [23] [24].

All operators use a structuring element (kernel) which has a pre-defined shape that is a subset of the real vector space $\mathbb{R}^2$, that is used to filter the input image, and changing the borders of each set of grouped pixels. The structuring element is usually denoted with the letter B.

- **Erosion**:

  The erosion operation is performed on a binary image to reduce or shrink it. Let A and B denote 2 binary sets in $\mathbb{E}^{\mathbb{N}}$, where $\mathbb{E}^{\mathbb{N}}$ is the N-dimensional Euclidean space, with elements a and b, respectively, where $a = a_1, a_2, ,,,, a_N$ and $b = b_1, b_2, ,,,, b_N$ being N-tuples of element coordinates. Then the binary erosion of A and B, shown in Figure 15(a), is the set of all elements x, for which $x + b \in A$, for every $b \in B$, and is defined as:

$$A \ominus_b B = \{x \in E^N | x + b \subseteq A, \quad for\ every\ b \in B\}$$

- **Dilation**:

  It is the process of combining two sets using vector addition. Formerly known as *Minkowski addition* after Hermann Minkowski. Let A and B denote 2 binary sets in $\mathbb{E}^{\mathbb{N}}$, where $\mathbb{E}^{\mathbb{N}}$ is the N-dimensional Euclidean space, with elements a and b, respectively, where $a = a_1, a_2, ,,,, a_N$ and $b = b_1, b_2, ,,,, b_N$ being N-tuples of element coordinates. The binary dilation of A and B, shown in Figure 15(b), is the set of all possible vector sums of pairs of elements, one coming from A and the other from B, and is defined as follows:

$$A \oplus_b B = \{c \in E^N | c = a + b, \quad for\ some\ a \in A\ and\ b \in B\}$$

- **Opening**:

The opening operation is obtained by first performing erosion on an image A using a structuring element B, and performing dilation on the output by using the same structuring element. The mathematical definition for such operation is:

$$A \circ B = (A \ominus B) \oplus B$$

This operation is used to remove all pixels that cannot contain the structuring element inside. Thus, the result of such operation is a smoother shape without bumps. Also if the input shape has narrow (thin) connections or protrusions, they are eliminated.

- **Closing**:

  The opening operation is obtained by first performing dilation on an image A using a structuring element B, and performing erosion on the output by using the same structuring element. The mathematical definition for such operation is:

$$A \bullet B = (A \oplus B) \ominus B$$

  This operation is used to fill the holes inside a bounding region that are smaller than the structuring element used. This also removes inward bumps.

3.3. Object Labelling

Connected object labelling is a task in computer vision and analysing patterns which helps detecting and labelling connected regions in a digital binary images, which has multitude of applications in different fields. For example, it is used in automated inspection, and analysing medical images [25]. It allows each pixel in a connected region to be grouped under the same label. However, it does not stop at binary images [26], as it can span higher dimensional image representations.

The task uses a small sliding kernel of two basic shapes, termed 4-connected neighbours and 8-connected neighbours. The kernel of choice is then slid over all image pixels (usually from top-left to bottom-right), giving adjacent region pixels, with relatively similar intensity values, the same label [27]. The result of the task is an image containing different regions, each has its own specific colour or greyscale value [28].

There are two ways for performing such task: *One component at a time*, and *Two-pass*.

One component at a time was originally made a part of Watershed segmentation algorithm. It goes through the image with number of times that is equal to the number of distant objects inside it.

Two-pass is the most used. From the name, it loops over all pixels in a binary image twice. The first is to assign temporary labels to all objects in the image. The second pass removes the extra redundant labels, and results in a component labelled image.

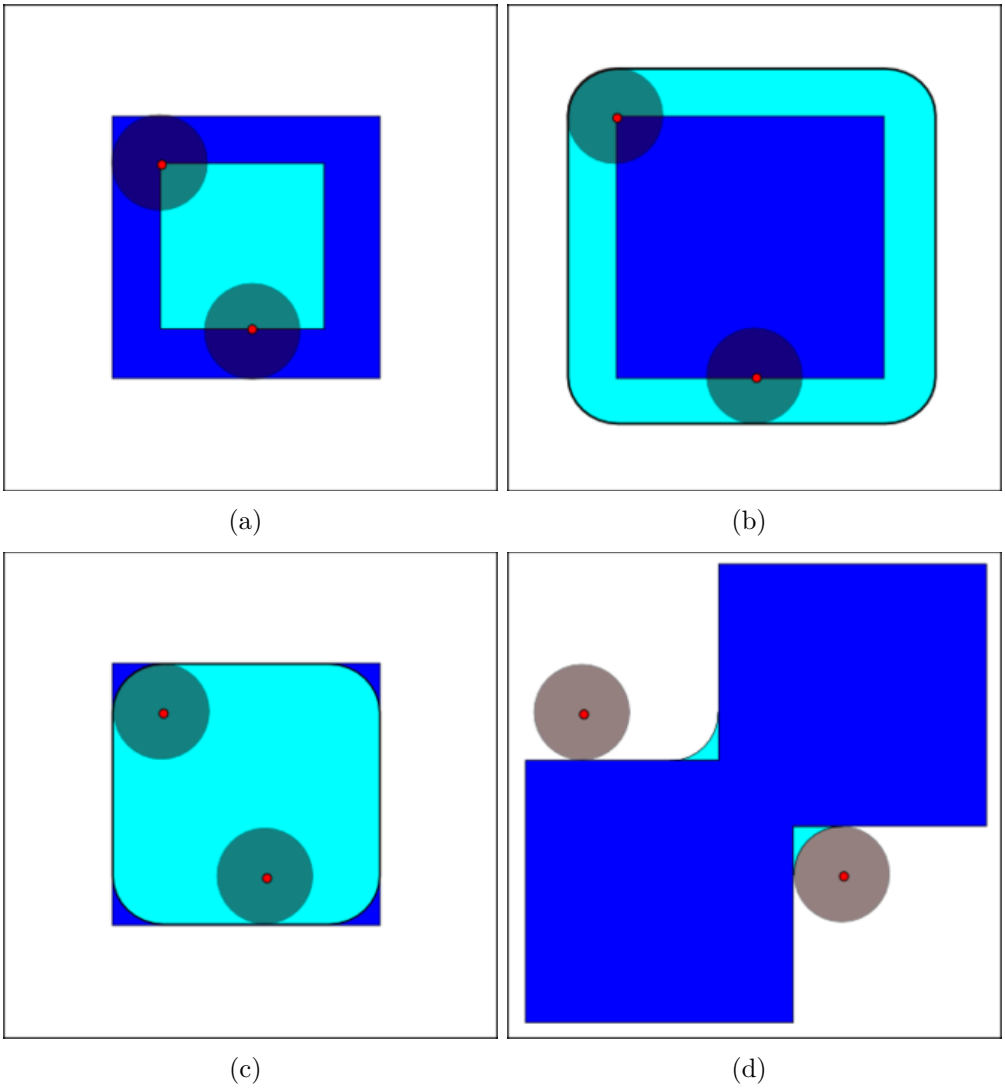
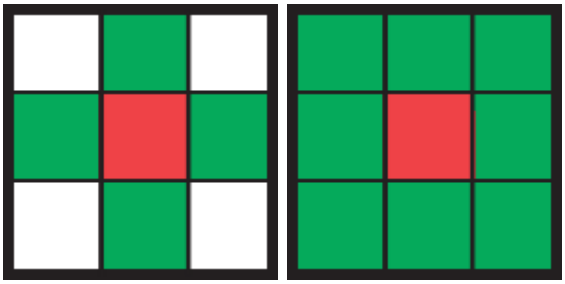


Figure 15. Examples of performing elementary morphological operations on the blue shapes, using a disk as a structuring element (kernel), that results in the cyan shape. (a) Erosion. (b) Dilation (c) Opening. (d) Closing.

3.4. Automation Examples

All the previous sections, serve as foundation for this section and the implementation to be built upon. Since the field of research concerned about studying the behaviour of aquatic organisms is relatively new, and it is using hardware that was mainly built for other purposes, as for example, underwater surveillance. Also the need for a standard pathway for analysing such input data became a necessity.

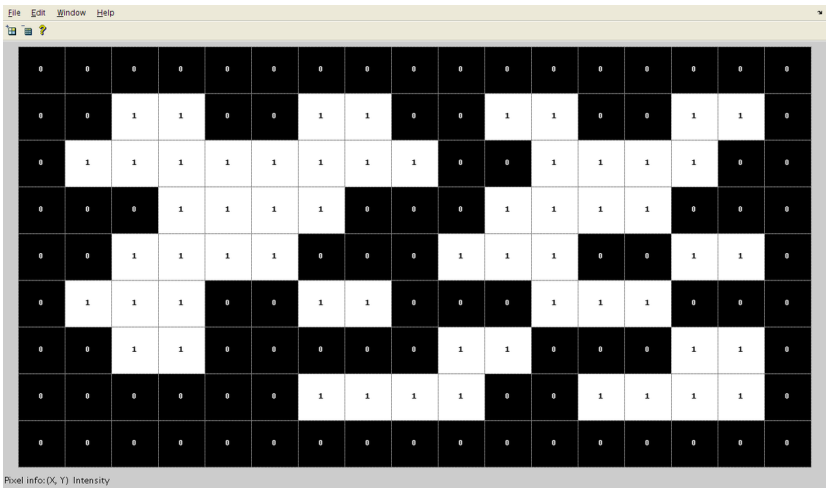
Such pathways, usually start with the part of static background subtraction [29], to remove unwanted samples and make it easier to identify foreground objects. However, the background subtraction process is not applicable for the mobile deployment of the SONAR, because the background would not be static anymore. Although, for the background subtraction to produce better results,



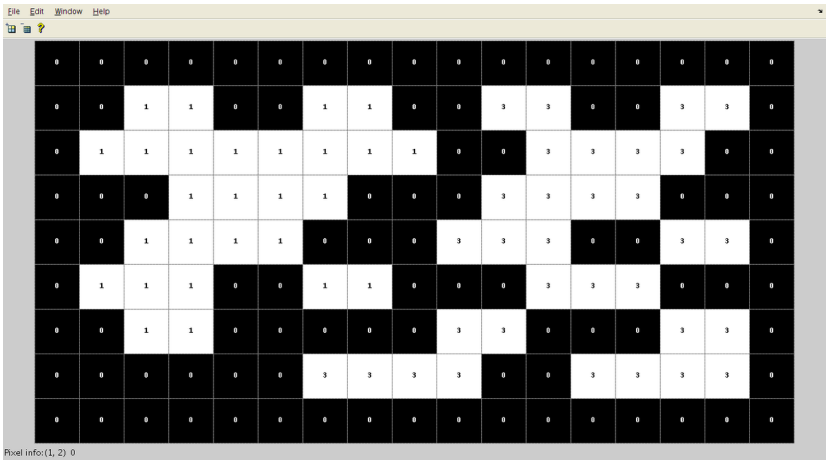
(a) (b)

Figure 16. Shape of the small sliding kernel. (a) 4-connected neighbours. (b) 8-connected neighbours. Each square resembles a pixel. The red square is the pixel in the middle is the pixel to be labelled. The green squares indicate which pixel is to be checked for connection.

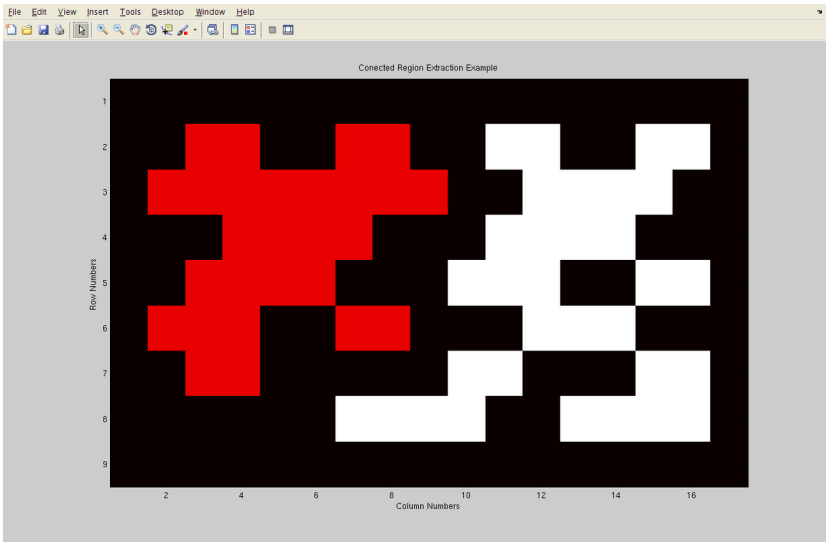
input frames should be smoothed to remove undesirable artefacts and detected noise. After the previous data cleaning processes, the resulting clean objects should also be processed to remove beam separation effects and their interference, which might appear in the form of holes inside the regions of the detected objects. All the previous processes are preprocessing steps for the object detection and tracking algorithms that can be later implemented.



(a)



(b)



(c)

Figure 17. Stages of connected component labelling algorithm. (a) Input binary image. (b) Output labelled binary image. (c) Output colour coded image.

4. IMPLEMENTATION

After a detailed description of the Hardware and its output data layout, and the brief overview on computer vision techniques in image processing, the foundation is now available on which a standard analysis pathway can be built upon.

4.1. File Operations

After acquiring the data, a library was developed (in Python) to perform read and write operations to the SONAR files. Files are written sequentially as a stream of bytes, and it has to be read accordingly. The library opens the file, reads the first 4 bytes, as an unsigned integer, that has file version encoded. File version values for each file type are mentioned in Table 1. Then based on the value of the version, the software continues reading sequentially packs of bytes, defining each pack to specific types. The definition for each pack of bytes, is mentioned in Table 5 and Table 6. The tables hold all possible headers for the latest file format (ARIS). Other older formats are just subsets of these headers.

This work provides a Python library that can do the operations mentioned in this chapter on SONAR files. This library can be used for development, where other systems can be built upon.

4.1.1. Read

The main file operation performed is reading. As the data is laid down in order, so it can be accessed by just knowing the order of the header or frame data needed, as follows:

$$\begin{matrix} Frame \\ Start \end{matrix} = \begin{matrix} FileHeader \\ Size \end{matrix} + \begin{matrix} Frame \\ Index \end{matrix} * \left(\begin{matrix} FrameHeader \\ Size \end{matrix} + \begin{matrix} Frame \\ Size \end{matrix} \right) \quad (1)$$

where *Frame Start* is the first byte in the requested frame, *File Header Size* is the number of bytes that the file header occupies, *Frame Index* is the number of the requested frame, *Frame Header Size* is the number of bytes that the frame header occupies, and *Frame Size* is the number of raw beams produced by the SONAR multiplied by the number of samples per beam, which gives the number of elements inside the captured image.

• **Example**: If working with ARIS file where *File Header Size* is 1024 Bytes and *Frame Header Size* is 1024, with the device producing 96 raw beams, and each beam has 1000 samples. The target is to jump to the place of the first byte in frame number 3, it can be calculated as follows:

$$FrameStart = 1024 + 3 * (1024 + 96 * 1000) = 292096$$

The result marks the first byte of the frame header from the requested frame number 3.

To be able to reconstruct images from a SONAR file, we have to read some attributes from file header and from each frame headers. According to [30], the needed attributes are listed as follows:

- Frame Count : Number of recorded frames inside the file.
- Raw Beams : Number of physical beams produced by the SONAR.
- Samples : Number of samples captured per beam.
- Sample Period : Time needed for sound wave to cross the length captured by a sample.
- Sound Speed : Sound speed in the SONAR ambient medium.
- Sample Start Delay : Time needed from pulse transmission to start sampling.
- Large Lens : Type of the mounted lens.

Yet there is one more attribute that is not written in the file itself, and it depends on the hardware of the SONAR itself. The property is the Field of View (FoV) and the Beam width. This can help in deciding where each beam angle starts, and where it ends.

4.1.2. Visualising Frames

From the acquired attributes we can deduce the parameters needed for reconstructing images and changing its format from Cartesian to Polar coordinates. Starting with calculating the *frame size*, and by multiplying number of raw physical beams by the number of captured samples across each beam. *Window start* length Equation 2 (i.e. the length between the SONAR unit and the first captured sample) and *Window length* Equation 3 (i.e. the length between the first and the last captured samples) are then calculated, which will help later in determining distances between detected fish and SONAR in metric units. *Window start* & *Window length* are shown in Figure 18(c).

$$\underset{Start}{Window}\ (m) = \underset{Delay}{SampleStart}\ (\mu s) * 10^{-6} * \frac{SoundSpeed(m/s)}{2} \tag{2}$$

$$\underset{Length}{Window}\ (m) = \underset{Period}{Sample}\ (\mu s) * \underset{Period}{SamplesPer}\ * 10^{-6} * \frac{SoundSpeed(m/s)}{2} \tag{3}$$

$$Range = WindowStart + WindowLength \tag{4}$$

$$\underset{Range}{Sample[i]}\ (m) = \underset{Start}{Window}\ (m) + \underset{Period}{Sample}\ (\mu s) * [i] * 10^{-6} * \frac{SoundSpeed(m/s)}{2} \tag{5}$$

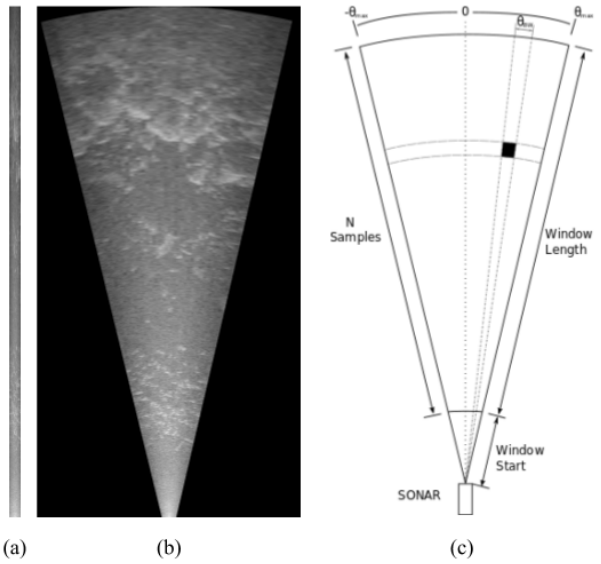


Figure 18. Image reconstruction from captured samples. Frame was captured in Vetsijoki river near the borders between Finland and Norway, on June 6th, 2016. ARIS operating at 48 Beams and 1994 samples per beam. (a) (48 * 1994) Raw frame samples as read from file, in Cartesian Coordinates. (b) (1032 * 1994) Reconstructed image, in Polar Coordinates. (c) Frame layout.

$$\begin{matrix} Sample \\ Length \end{matrix} (m) = \begin{matrix} Sample \\ Period \end{matrix} * 10^{-6} * \frac{SoundSpeed(m/s)}{2} \tag{6}$$

Also *Sample Length* is calculated from Equation 6 (i.e. the length covered by 1 sample), and it also indicates the resolution of the captured image. While the value of *Sample [i] Range* is the length between a specific sample i and the SONAR.

One value is left to be defined, which is the maximum angle θ_{max}, denoting the rightmost angle of the horizontal Field of View. It is device-dependant, as each device has its own FoV value.

The purpose of calculating all the previous attributes is drawing the outline of the captured data in Real-World (Polar) coordinates in Figure 18(c). This eases the task of converting the captured raw data samples from Cartesian coordinates in Figure 18(a) to the human-friendly real-world Polar coordinates in Figure 18(b).

4.1.3. Write

Since the file format has some user specific fields, any developed software should have the ability to write to those fields. These fields does not have any specific description or regulations on how they should be used. However, this work assumes the following `User-Defined` bytes in the frame header should be used to store number of fish found in each frame. While `User ID` bytes in the file header

should contain IDs of the personnel responsible for analysing captured footage. In the `Header ID` field in the file header, there should be some simple comments by the analyser. However, any `reserved` field in any header, should never be written to, to allow for backward compatibility in the future.

4.2. Proposed Framework

After redrawing raw data into Polar coordinates, it is time for the generated imagery to be processed. Here the need for a standardised analysis scheme rises. In this section, a standard analysis framework or scheme is introduced for processing SONAR files. The framework is shown in Figure 19.

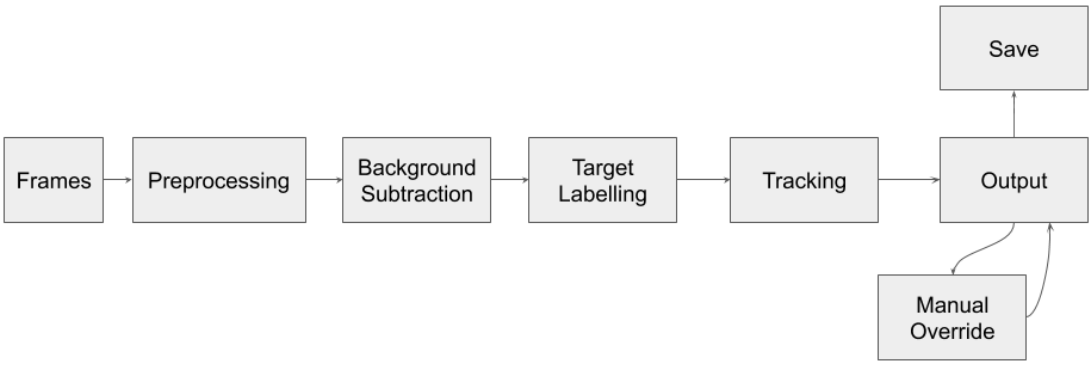


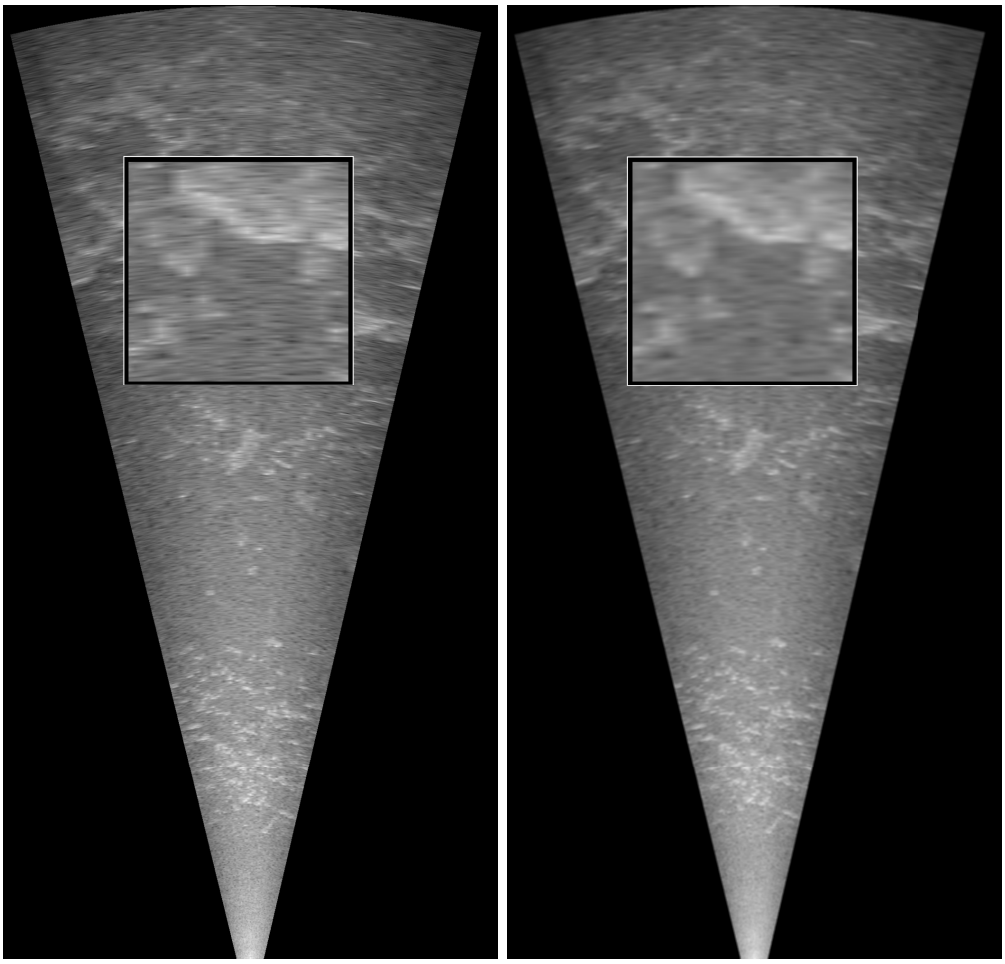
Figure 19. Generic flow diagram of proposed pathway for analysing SONAR data.

4.2.1. Preprocessing

Input frames has to be preprocessed before being analysed. It is used to reduce captured noise. This work uses a simple spatial smoothing in the preparation stage. Blurring, shown in Figure 20, helps to increase the efficiency of background subtraction and object detection in the upcoming steps, as it removes unnecessary small edges and random brightness variations.

4.2.2. Background Subtraction

As mentioned earlier in section 3.1, this technique is used to separate moving objects in a sequence of captured frames. This work uses one of the recursive background subtraction models, which is Mixture of Gaussians (MoG). The algorithm models each background pixel by a mixture of K Gaussian distributions. The number K is pixel-specific. Hence, each background pixel has its own K, which increases adaptability and robustness against brightness changes. However, the legacy MoG uses one K value for all pixels. Figure 21(b) shows the output mask from background subtracting 21(a). Although the source



(a) (b)

Figure 20. Blurring input frame. (a) Original input frame. (b) Blurred output frame. same areas are maximised to about twice their size to show the effect of blurriness.

has only one fish, the mask output of background subtraction has 3 objects, where 2 of them are noise that need to be removed.

4.2.3. Morphological Operations

The output of background subtraction step is usually not the cleanest. The output mask sometimes contains holes and protrusions, aside from the small extra noise that was mistakenly identified as moving objects. Morphological operations can solve such problems. The main morphological operators are mentioned in Chapter 3. In this work, closing and opening operations are performed respectively. These operations close the opened holes inside closed mask regions, and remove small noise masks. In Figure 21(c), the output of the morphological operations contains only one object compared to 21(b) which has 3 objects including noise.

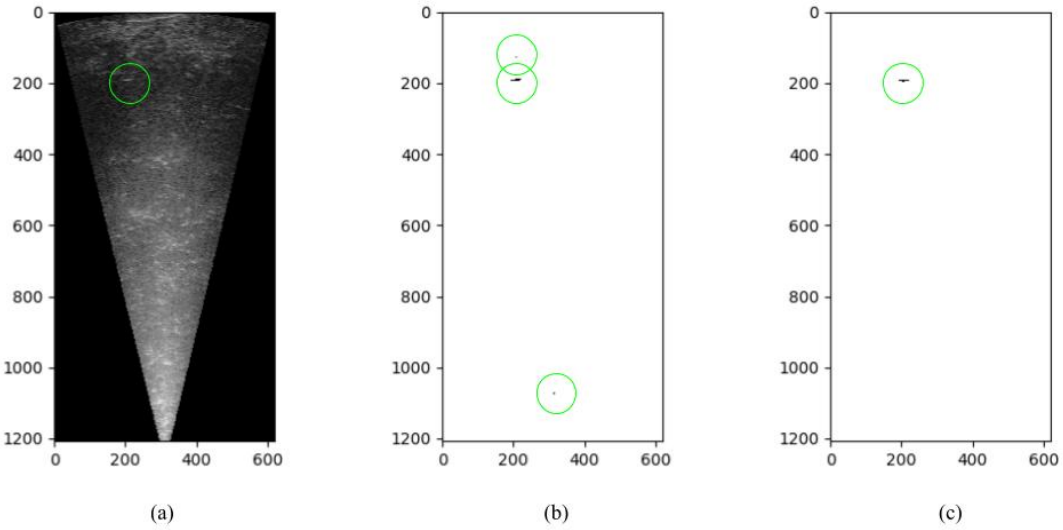


Figure 21. Frame from data captured in Tornionjoki, on the borders between Finland and Sweden. SONAR is covering a distance of 27.3 meters, using 96 beams, sampling 1208 samples per beam. Size of the input frame (96*1208), size of the real-world polar coordinates image (621*1208) (a) Original image. (b) Background Subtracted image. (c) Morphed output image.

4.2.4. Connected Object Labelling

After obtaining a clean mask, it is the perfect stage for object detection. Since each object is already separated, each object can be determined by using Connected Object labelling technique. The algorithms for such task was mentioned previously in connected object labelling section. Figure 22(b) shows the 2 detected fish using connected component labelling from Figure 22(a). This result will be used in the tracking section.

4.2.5. Tracking

There are various tracking algorithms in the literature. The most used of all, is Kalman filtering, which was used in [31]. The method predicts the future observation according to previous measurements. However, for this implementation, a much simpler algorithm is used.

After the objects are labelled, each is assigned a new ID, and its current location is saved. The next frame is loaded, and same processing is performed. When the frame is clean enough and objects are visible, each is assigned a new ID. Each 2 consecutive frames are compared together to check if any 2 objects are in a radius R from each other from the both frames. If there is an object's centroid in the newer frame near the object's centroid from the older frame, the newer object is assigned the same ID as the older object.

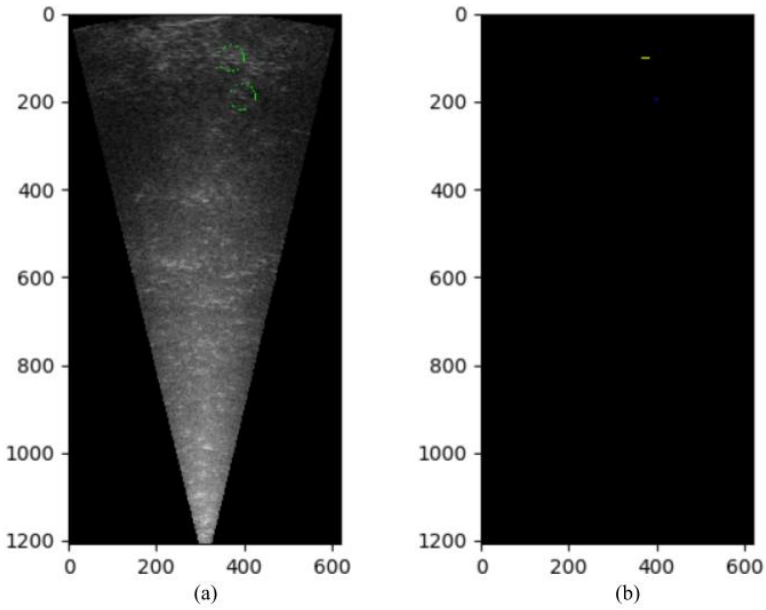


Figure 22. Connected object labelling operation. (a) Original image. (b) Connected component labelled image.

From Figure 23, objects from frame *i+1* are compared to objects from frame *i*. The green object with *ID=v* is in the search radius *R* of the cyan object from frame *i* with the *ID=u*. Thus, it is assigned the same *ID*, so it is changed from *v* to *u*. However, the red object with *ID=w* is listed as a new object.

The program deals with detected fish (i.e. objects) as a stack of 3 stages. The first stage is when the fish is just detected, shown in Figure 24 as the lowermost level. To avoid mistakenly detected rogue noise as fish, the detected fish has to stay visible on screen for *n* number of frames to be considered as fish. Usually, for a fish to swim all the way upstream, it takes tens of frames with moderate frame-rate. After making sure that the detected object is actually a fish, it moves to the next level of the stack, shown as the middle layer in Figure 24, and stays there for as long as it is on the screen. Meanwhile, the program keeps track of all the positions that the fish has been throughout its lifetime on the screen. After the fish disappear from the screen for more than *m* frames, it is moved to a storage, shown in Figure 24 as the archive layer, until the program finishes and it is exported as the result from the analysis session.

This simple tracking algorithm has three main downsides, which can affect the result. The first problem is when there are multiple fish swimming alongside each other as shown in Figure 25(a), the tracking algorithm will mix them together, which reduces the number of the output fish relative to the number of fish captured in the file.

The second problem, is that fish might sometimes swim behind an occlusion as shown in Figure 25(b). Tracker stops at the point before the fish becomes occluded and never reaches the other point where the fish gets out after the occlusion.

The third problem, is that some fish swim in intertwined paths as shown in Figure 25(c). The tracker in these situations calculate fish more than one

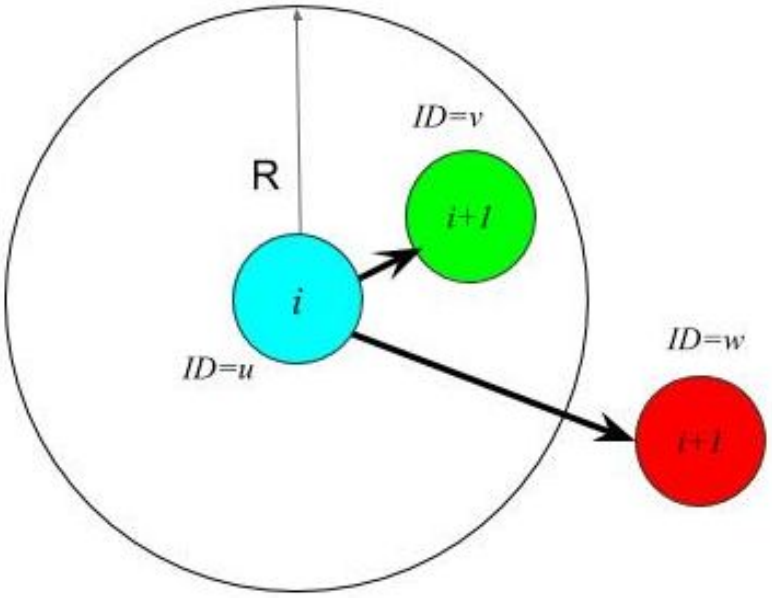


Figure 23. Tracking model used. R indicates the search radius around the detected object in cyan with $ID{=}u$ in frame number i. The red and green circles indicate 2 detected objects in frame $i{+}1$ with $ID{=}w$ and $ID{=}v$ respectively.

time, and sometimes mixes the two tracks producing very sharp edges in the fish swimming path.

4.2.6. Output and Manual Override

After finishing the analysis process, results from the archive shown as the uppermost level in Figure 24, are retrieved and exported into various formats.

The legacy format which is produced by Sound Metrics' software is a text file, with the data written inside as a template. The problems with such format is the limitation of integration with new technologies. For instance, contemporary systems, include but not limited to, `JSON`, `XML` or `CSV`, as most modern programming languages have interfaces to read and write to such files, which eases re-usability. Also one of the limitations of the legacy system is that it refers to the whole path of the detected fish as one reading, which is considered waste of valuable information.

The software developed in this thesis is able to produce the result in two of the most modern file formats, `JSON` and `XML`. Also, producing text templates that is compatible with the legacy systems is easily done. However, the benefit of having such new formats, allows for adding extra information into the results. For instance, all the frame numbers that a fish has been detected and the dimensions of each fish can be recorded and saved in the output. A benefit of such information is the ability for a program and/or human viewing the data to only display frames with the detected fish to save time and lower processing costs. On the other hand,

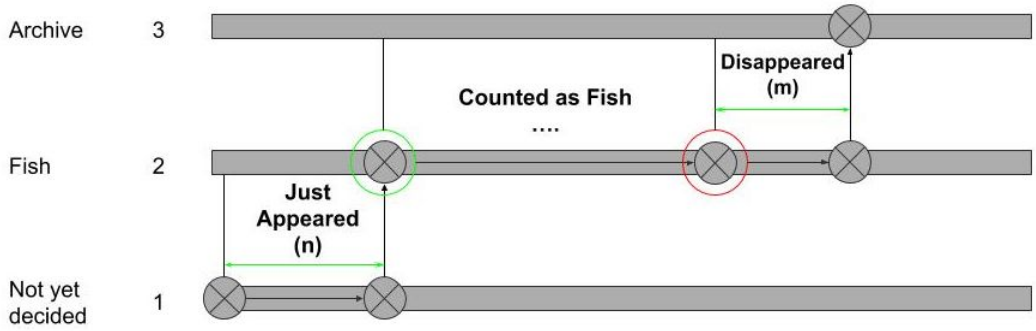


Figure 24. Simple tracking algorithm used to detect fish swimming in front of a SONAR unit. Diagram shows 3 layers. A detected object goes first to the lowermost layer. If proven to be a fish, it goes to the middle level, and stays there through its on-screen lifetime. When it disappears, it goes to uppermost level, and archived for later.

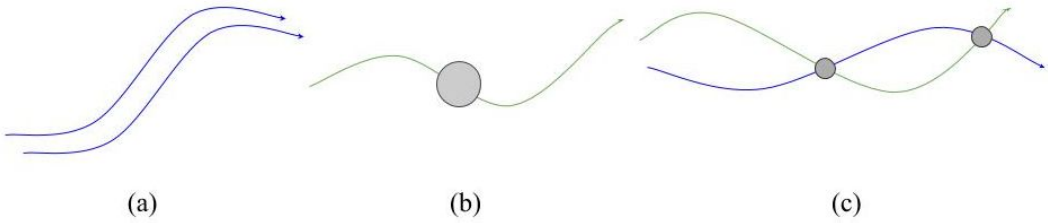


Figure 25. Arrows showing fish swimming patterns. (a) Two fish swimming near each other. (b) One fish swimming behind an occluding body. (c) Two fish swimming in intersecting paths.

legacy formats only saves one frame that the observer found the fish. This is only useful for counting, but not for anything else.

As previously mentioned, the software can save the frames where each fish was visible. This provides a new feature, which is *Manual Override*. The feature allows the observer to view the results of processing a file, and view the detected fish and its path. In case the software has falsely identified noise as fish, this can be corrected through manual override.

4.3. User Interface

Now the whole library to read, write and analyse files has been designed. However, this is only developer friendly, while the purpose of the project is make a user-friendly software to help non-technical users to view and analyse captured data.

This section introduces a simply designed user interface, shown in Figure 26, that interfaces the programmed library to interact with SONAR files. It was designed using Qt tools, and it is also available as open-source for development.

As mentioned before, there is an analysis pathway that should be followed whenever designing an analysis software, shown in Figure 19. Therefore this software follows those steps.

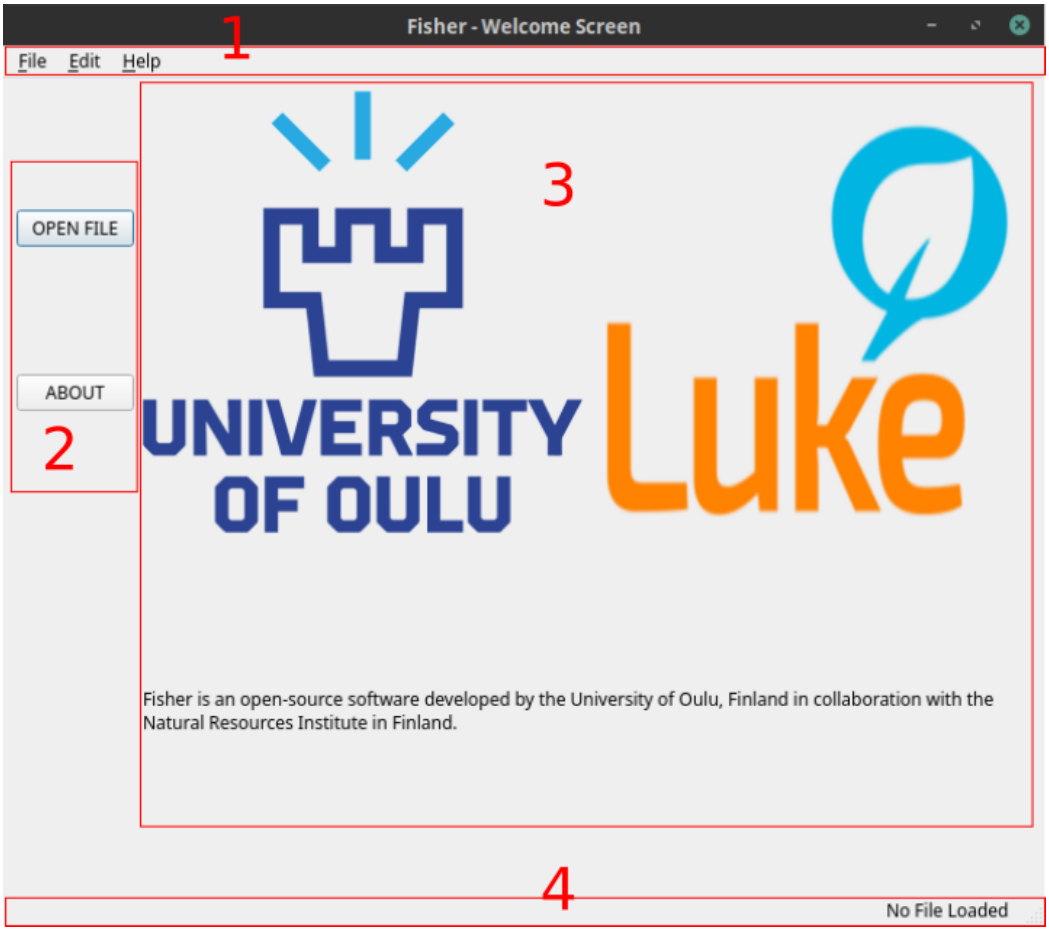


Figure 26. Simple user interface - Welcome Screen. The title bar shows the name of the program *Fisher*. This is the first thing that the a user sees whenever they start the program. (1) Menu bar. (2) Quick Access. (3) Information Section. (4) Status bar.

4.4. Work-flow

After launching the program, the user is greeted with a welcome screen, that shows some information about the university of Oulu and LUKE. From the welcome screen, the user can open files for analysis by using the *OPEN FILE* button. After choosing the file to analyse, the file is opened in the *Viewer* view, shown in Figure 27. This view helps the user to skim through captured frames of the chosen file, while showing the user real-world coordinates (distance: is the distance in meters between the SONAR unit and the mouse cursor position, angle: is the angle from the vertical centre of the shown frame and the mouse cursor position) in the status bar when the mouse hovers over any part of the image. The status bar also shows the index of the displayed frame relative to the total number of frames in the file. The frames inside the file are zero-indexed. However, in this part it is one-indexed to make it more human friendly.

The user can use playback options, as the slider goes directly to a specific frame. While the *Next* and *Previous* buttons go forward and backward, respectively, one

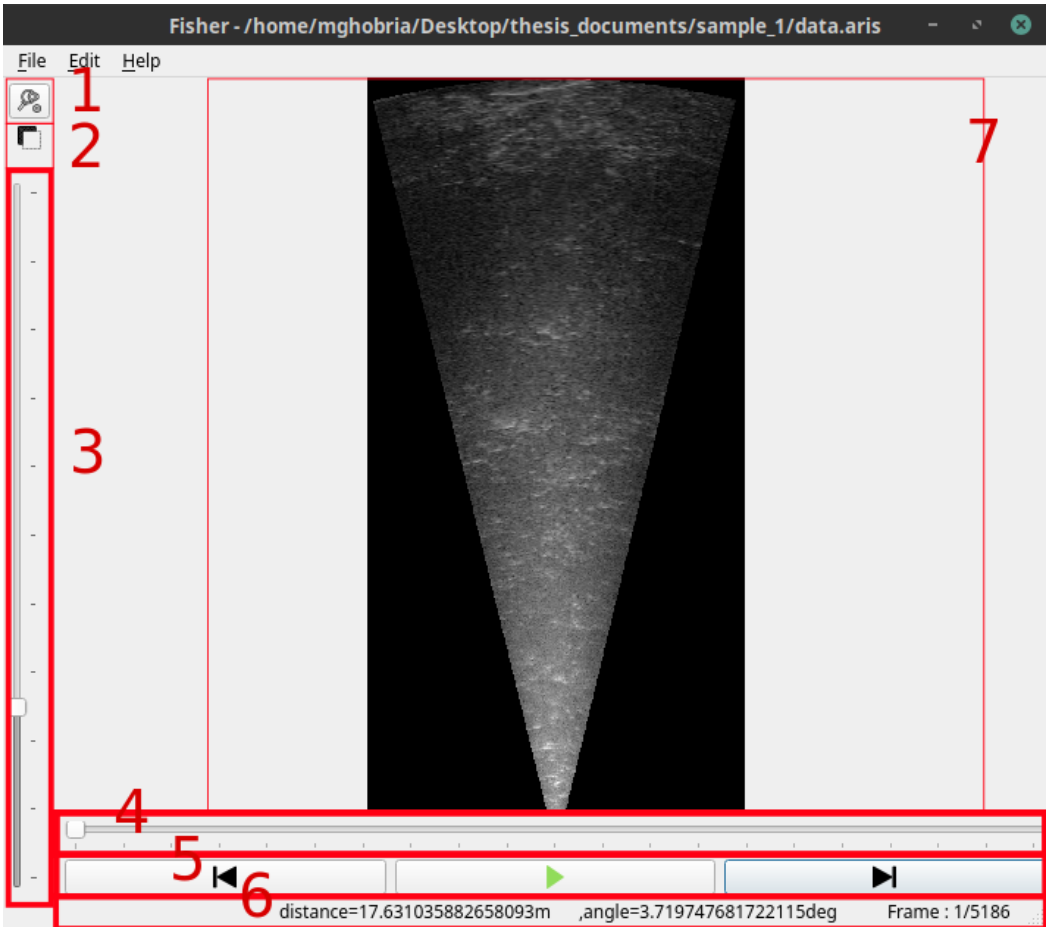


Figure 27. Simple user interface - Viewer. The title bar shows the name of the program *Fisher*, followed by the path of the opened file. (1) Auto-analyser button. (2) Background subtraction button. (3) Background subtraction threshold controller. (4) Frames slider. (5) Playback control buttons. (6) Status bar. (7) Frame viewer, that shows the frames from the opened file.

frame at a time. The *Play* button is used to play the frames back using the quickest playback speed suitable for the machine.

The *Background subtraction* button is used to show another instance of the current frame, next to the original version of the current frame, with no background, as shown in Figure 28. This makes it easier for a human to spot any moving objects while skimming through the frames. The slider beneath the button controls the intensity of the subtracted background. If too high, the program removes everything even the moving objects. If too low, the program does not remove most of the noise and it corrupts the view. This feature's settings do not affect the auto-analyser settings.

The auto-analyser tool is the most important feature in this piece of software highlighted in Figure 27 as number 1. Whenever the button is pressed, the tool's settings pop up in a new window, as shown in Figure 29.

The tool's settings control how the analyser analyses the input frames. All the settings shown in Figure 29 are set to a normal default. When the analyser starts,

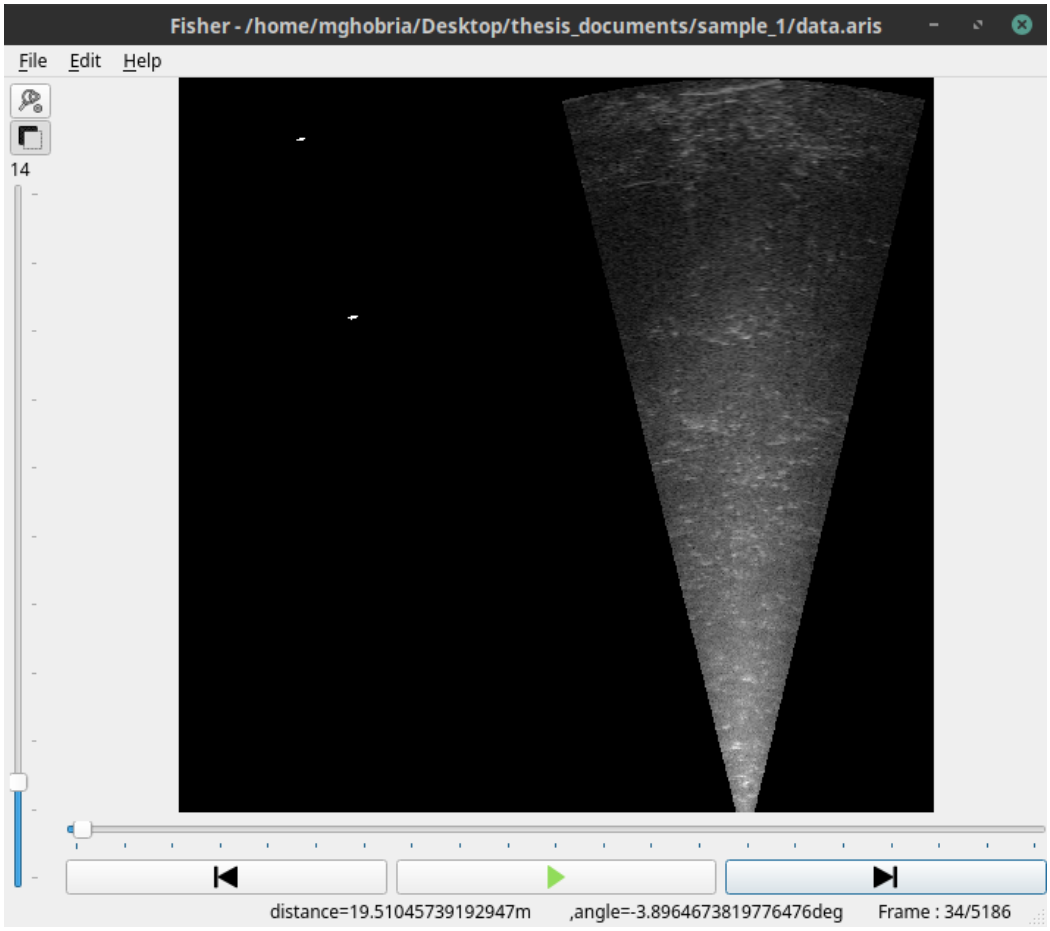


Figure 28. Simple user interface - Viewer no background. This view appears when a user clicks on *Background subtraction*. The left side shows the frame from the right side but with no background elements.

it shows a new window where the frames are shown while being processed, and the moving fish are marked and shown.

4.5. Output

After finishing the analysis, the viewer window is shown again, but this time with a new column containing the analyser results. The column contains a list of items. Each item is a fish result, and each result can be viewed and marked in the viewer.

The window, shown in Figure 30, enables the user to manually override the detected fish. After a user views the results, one should mark the list item as fish or leave it blank for no fish. When the user presses the *Export* button, only the list items that were marked as fish are saved to an output file.

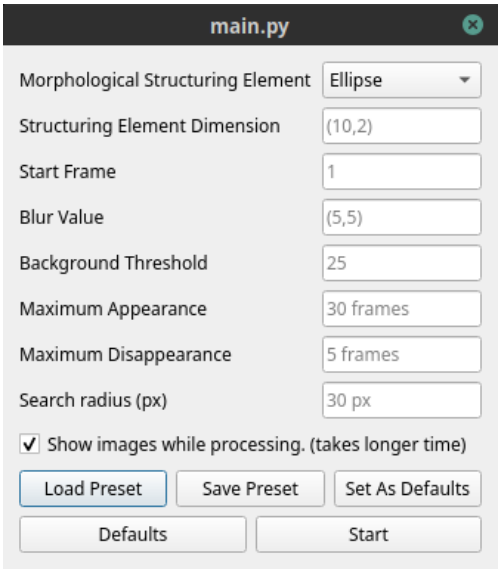


Figure 29. Auto-analyser tool's settings window.

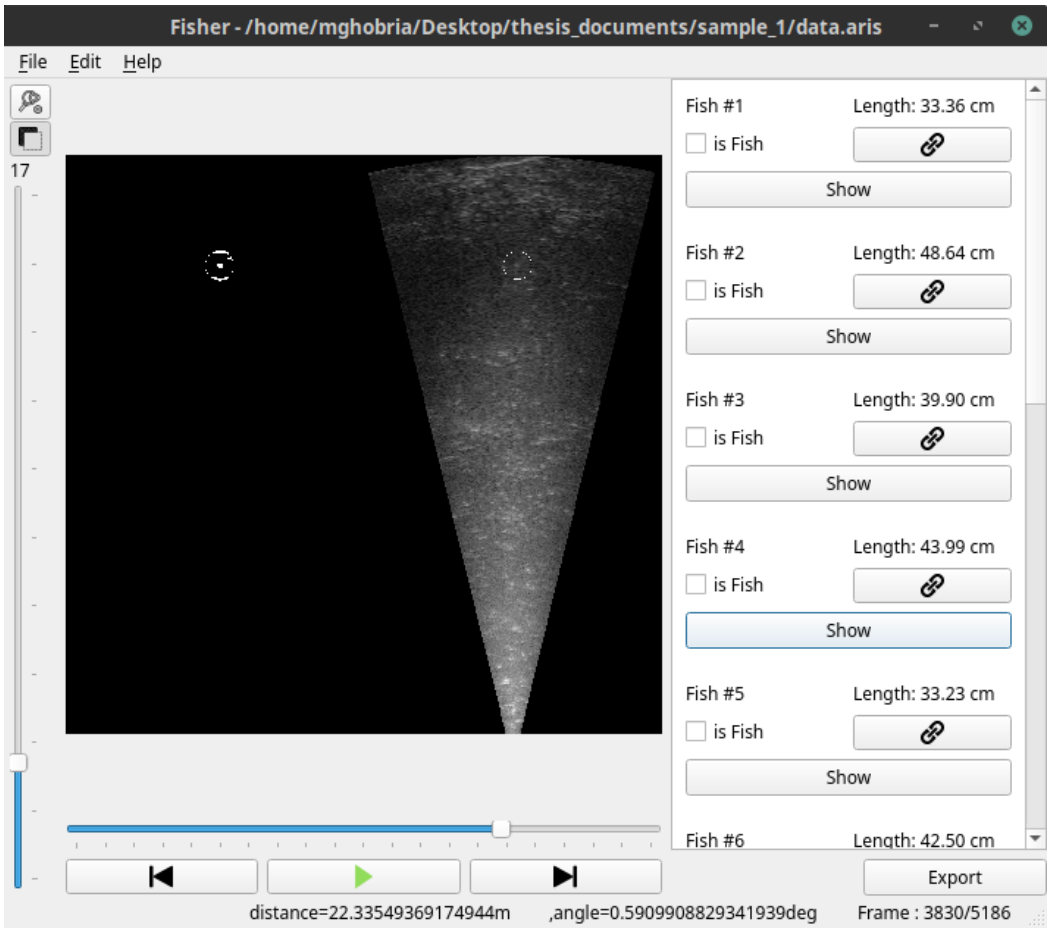


Figure 30. Auto-analyser output.

5. EXPERIMENTS & DISCUSSION

5.1. Tuning of Variables

This work allows users to freely tune and tweak analysis parameters if needed. Available parameters are shown in Figure 29. The morphological structuring element is the kernel used to perform the needed morphological operations of closing and opening to fill the holes in the detected objects and removing any protrusions. This kernel, by default, is set to an *Ellipse* which is considered the closest shape to the fish shape to be detected. Its dimensions have been set to a small shape to cope with the low resolution images captured by the SONAR unit.

Start frame value enables the user to analyse only a part of the file while ignoring the rest of the frames. This could be useful if there is some unwanted transient noise in the beginning of the recording to be analysed.

Blurring is one of the preprocessing tasks, to smooth the image. This value controls the dimensions of the smoothing matrix. The matrix locates itself on each pixel in the image and averages it according to the neighbouring pixels included into the blurring matrix. Hence, the bigger the matrix, the smoother the image will be.

The background threshold is the parameter that controls the behaviour of background subtraction module. It is applied to the squared Mahalanobis distance between the background model and the pixel in question [32] [33] [34] [35], to decide whether it is a background or a foreground pixel. When this value is chosen to be small, it increases processing time dramatically. If this value becomes too high, it labels most of the pixels as background pixels, reducing the capability of the program to detect fish.

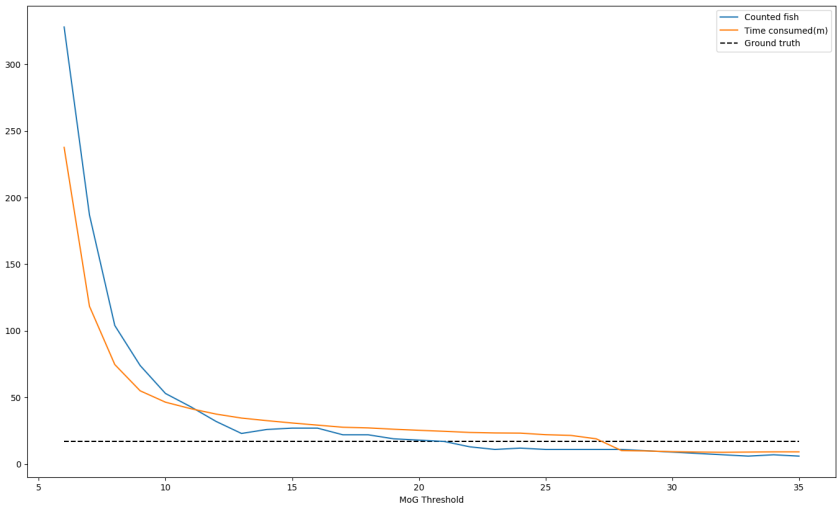


Figure 31. Comparison between different *Background threshold* values, performed on the same dataset.

A sweep on the *background threshold* has been performed, using LUKE's dataset. Starting from value 6 to 35, a file with 5186 frames has been analysed for each single value while calculating the time of each run (in minutes). Results are shown in Figure 31. These results are produced using a computer with a 2.0 GHz, Intel i7 processor. The manual count of fish in those frames has been considered the ground truth, shown as a black dashed line. The blue line shows the amount of detected objects to be considered as fish. The yellow line shows the time consumed to analyse the file at each run.

From this experiment, the ideal values of the threshold should be the ones that produce slightly more objects than the ground truth, which means that it detects more objects which can be filtered later.

The maximum appearance and minimum disappearance form a kind of hysteresis loop, such that an object is considered a fish if it has been on the recording for a number of frames more than the minimum. The fish is then discontinued from tracking when it disappears for a number of frames more than the maximum.

The search radius is the parameter that controls the search area around a detected object to determine it's position in the new frame and link it to the previous frame measurement. If this value is high, it might make tracking output worse, because it might link distant swimming fish as the same current fish. If the value is too small, it might discontinue counting the current fish.

5.2. Analysing Batches

To test the performance of this piece of software, 37 files were analysed each of which is about 2 GB in size and contains more than 20 thousand frames, and the result of the analysis is shown in Table 3. This analysis was done on a relatively high background threshold value, which resulted in producing slightly lower counts than the ground truth. However, from the results we can notice the high jumps in the number of counted fish with the files that were captured to monitor smaller ranges. Since, smaller ranges lead to higher resolutions, therefore, the algorithm produced more counts that were missed by the manual human eye.

As seen from Table 3, analysed files captured from the same river, Vetsijoki, but they are divided into three different data sets according the window range that they cover. The first is a long window range, spanning about 38m, which was captured in the beginning of June, 2016. The second, has a shorter window range, spanning about 11m, which was captured in August, 2016. The third, has a marginally, yet effective, shorter window range, spanning slightly less than 11m, which was captured in middle of June, 2016. In the data sets with higher window ranges, the program often produces lower results than manual counting (i.e. the ground truth), this is due to to lower resolution, where a fish might span lower number of pixels compared to other higher resolutions.

When the resolution upsurges, like in the second data set, the number of detected fish starts rising, compared with the ground truth. It reaches it's highest in the third data set, which covers the least window range.

However, in the third data set analysis result, it is clearly visible that the output is much higher than the ground truth in some cases. The output of these samples from the data set was visualised to prove that it has detected more fish that was missed by manual tracking. When considering the manual tracking as the ground truth, the samples with shorter window ranges achieve as high as 100% accuracy. While the overall accuracy of the system is ($\approx$ 57,3%).

5.3. Exporting Results

The most extensible format in any development framework right now is `JSON`, and it has been used in this work to produce the output. The template of the output is shown in the next code snippet. It has the number of keys the same as the number of fish detected. The output also includes the processing time and date, alongside the parameters used in the analysis process, just in case a user needed to replicate the results.

Each fish is recorded as a set of frames that it has appeared in, and also its location in those frames. In addition to the locations, which marks the centroid pixel for each fish, we also have the bounding boxes for each fish, which are saved as top, left, width, and height. This allows a full redraw or extraction for the fish from the file, for visualising the results.

One of the biggest upsides for using such format for the output, is the integration. Hence, any entity that uses the software can integrate the results with its pre-existing systems.

5.4. Future Work

Since the field is quite modern, the number of conducted researches is relatively low. However, the topic and the target from the research are becoming more relevant every day with the need for underwater surveillance and aquatic creatures stocking for preserving threatened species.

This software is still not robust nor fail-proof, but with the right modifications and testing it will easily reach that. The tracking algorithm used is simple and can be improved to higher grade one, for instance, Kalman filtering may be used to predict the position of the fish in the new frame, to make the search and measurement area easier. Also, some computer vision techniques should also be used to make sure that the software avoids the problems mention in Figure 25. A solution to the fish crossing paths problem may be by using smoothing function that chooses the next fish location to make the path as smooth as possible.

The software also needs a verification system, that can analyse the detected object, to verify if the object is fish or not. The first idea that comes to mind is to train a machine learning algorithm that can be used to classify. Luckily, this software can make the training easier, because it already provides the results to human as images so the user can decide. This time when the user decides and saves the output, it becomes a labelled set of data to use for training.

Table 3. Automatic and Manual counting results comparison

Automatic	Manual	Window Start (m)	Window End (m)	Window Range (m)
3	4	1.67	40.28	38.61
0	4	1.67	40.28	38.61
2	3	1.67	40.27	38.61
1	2	1.67	40.27	38.61
1	8	1.67	40.27	38.61
2	4	1.67	40.28	38.61
3	8	0.67	11.69	11.02
4	10	0.67	11.69	11.02
4	6	0.67	11.69	11.02
7	9	0.67	11.69	11.02
7	9	0.67	11.69	11.02
0	3	0.67	11.69	11.02
2	5	0.67	11.69	11.02
0	1	0.67	11.69	11.02
1	0	0.67	11.69	11.02
2	0	0.67	11.69	11.02
1	1	0.67	11.69	11.02
2	6	0.67	11.69	11.02
8	6	0.67	11.69	11.02
1	4	0.67	11.69	11.02
8	7	0.67	11.69	11.02
4	5	0.67	11.69	11.02
5	6	0.67	11.69	11.02
0	2	0.67	11.69	11.02
5	12	0.67	11.69	11.02
4	12	0.67	11.69	11.02
3	7	0.67	11.69	11.02
5	4	0.67	11.69	11.02
3	8	0.67	11.69	11.02
6	9	0.67	11.69	11.02
0	3	0.67	11.69	11.02
2	3	0.67	11.69	11.02
1	1	0.67	11.6	10.93
3	3	0.67	11.6	10.93
6	5	0.67	11.6	10.93
16	6	0.67	11.6	10.93
24	4	0.67	11.6	10.93

6. CONCLUSION

This work has gone through the topic of counting and stocking fish swimming through SONAR spans placed underwater, which is one of the measures taken by countries, organisations and entities to preserve aquatic creatures, especially Atlantic Salmon (Salmo Salar) which is considered an endangered species.

The history of development of SONAR devices was laid out since its start. Explaining categorisation schemes based on their mode of operation and the type of transmission, beginning with single-beamed device, all the way to multi-beamed devices. Examples of such devices are the DIDSON and ARIS, which this work discusses thoroughly, justifying the reasons for using these devices.

The composition of such devices was illustrated, and the theory of operation has been discussed, providing some examples. Then, the modality of image formation explains how these devices produce video-like output that can be further analysed.

Going further, this work showed how these devices were set up in their operation locations covering river flows to increase the efficiency of their output. The general format of the output files was also illustrated.

Next, in the computer vision chapter, all the needed background knowledge has been established by explaining each of the processes, algorithms or tasks that will be used. These concepts have been followed mainly through the implementation while designing the proposed framework, and the user-friendly software for the users.

Trying the software led to acceptable results, which were mentioned in the Experiments chapter, which proves that the software has potential to it, with some tweaking and polishing, it will be robust and fail-proof.

All in all, the benefits of an automated system for analysing SONAR files can be seen. It reduces time and effort consumed by personnel to analyse such files. The task is tedious and tiresome.

The absence of a robust analysis pathway is holding this area of research backwards. However, the proposed pathway paves the way for development in the field, allowing future development to build upon the current work.

7. REFERENCES

[1] Fisheries N. (2018), Atlantic Salmon - Protected | NOAA Fisheries. URL: https://www.fisheries.noaa.gov/species/atlantic-salmon-protected.

[2] Angeles M.A..E.P.A.P. & Us W..P..C., The Salmon Life Cycle - Olympic National Park (U.S. National Park Service). URL: https://www.nps.gov/olym/learn/nature/the-salmon-life-cycle.htm.

[3] Species Profile for Atlantic salmon (Salmo salar). URL: https://ecos.fws.gov/ecp0/profile/speciesProfile?spcode=E07L.

[4] Irvine J.R., Gross M.R., Wood C.C., Holtby L.B., Schubert N.D. & Amiro P.G. (2005) Canada's species at risk act: An opportunity to protect endangered salmon. Fisheries 30, pp. 11–19.

[5] Jenkins J.G. (1974) Nets and coracles. London, David & Charles.

[6] Parrish D.L., Behnke R.J., Gephard S.R., McCormick S.D. & Reeves G.H. (1998) Why aren't there more atlantic salmon (salmo salar)? Canadian Journal of Fisheries and Aquatic Sciences 55, pp. 281–287.

[7] Powles H., Bradford M.J., Bradford R., Doubleday W., Innes S. & Levings C.D. (2000) Assessing and protecting endangered marine species. ICES Journal of Marine Science 57, pp. 669–676.

[8] Fisheries N. (2019), Recovery Plan (2019) for the Gulf of Maine Distinct Population Segment of Atlantic Salmon (Salmo salar) | NOAA Fisheries. URL: https://www.fisheries.noaa.gov/resource/document/recovery-plan-2019-gulf-maine-distinct-population-segment-atlantic-salmon.

[9] Monitoring of salmon runs in River Tornionjoki and Simojoki. URL: https://www.luke.fi/en/natural-resources/fish-and-the-fishing-industry/fish-resources/salmon-2/monitoring-of-salmon-runs-in-river-tornionjoki-and-simojoki/.

[10] Atlantic salmon anatomy. URL: https://www.atlanticsalmontrust.org/wp-content/uploads/2016/12/atlantic_salmon_anatomy.pdf.

[11] Hansen R.K. & Andersen P.A. (1993) 3d acoustic camera for underwater imaging. In: Acoustical Imaging, Springer, pp. 723–727.

[12] Smyth C., Poynton F. & Sayers J. (1963) The ultrasound image camera: Proceedings iee, 110, no. 1, p. 16 (1963). Ultrasonics 1, p. VII.

[13] Wille P. (2005) Sound Images of the Ocean: in Research and Monitoring. Springer Science & Business Media. Google-Books-ID: r5CUczPBlp0C.

[14] Chernyak V.S. (1998) Fundamentals of Multisite Radar Systems: Multistatic Radars and Multistatic Radar Systems. CRC Press. Google-Books-ID: 0bEU2oBzMJEC.

[15] Belcher E., Hanot W. & Burch J. (2002) Dual-Frequency Identification Sonar (DIDSON). In: Proceedings of the 2002 Interntional Symposium on Underwater Technology (Cat. No.02EX556), pp. 187–192.

[16] Moursund R.A., Carlson T.J. & Peters R.D. (2003) A fisheries application of a dual-frequency identification sonar acoustic camera. ICES Journal of Marine Science 60, pp. 678–683. URL: https://doi.org/10.1016/S1054-3139(03)00036-5.

[17] dfg.webmaster@alaska.gov, Sonar Tools: DIDSON, Alaska Fisheries Sonar, Alaska Department of Fish and Game. URL: http://www.adfg.alaska.gov/index.cfm?adfg=sonar.didson.

[18] Sound Metrics - About Us. URL: http://www.soundmetrics.com/About-Us.

[19] Belcher E., Lynn D., Dinh H. & Laughlin T. (1999) Beamforming and imaging with acoustic lenses in small, high-frequency sonars. In: Oceans '99. MTS/IEEE. Riding the Crest into the 21st Century. Conference and Exhibition. Conference Proceedings (IEEE Cat. No.99CH37008), vol. 3, IEEE & Marine Technol. Soc, Seattle, WA, USA, vol. 3, pp. 1495–1499. URL: http://ieeexplore.ieee.org/document/800215/.

[20] EdgeTech the leader in underwater technology. https://www.edgetech.com/pdfs/ut/app_note_beamwidth.pdf. Accessed: 2019-05-02.

[21] Juha Lilja Mikko Leminen A.R.S.S.S.P. (2013), Monitoring of salmon spawning run in the river tornionjoki.

[22] Sen-Ching S.C. & Kamath C. (2004) Robust techniques for background subtraction in urban traffic video. In: Visual Communications and Image Processing 2004, vol. 5308, International Society for Optics and Photonics, vol. 5308, pp. 881–893.

[23] Shih F.Y. (2017) Image processing and mathematical morphology: fundamentals and applications. CRC press.

[24] Haralick R. (1981) Some neighborhood operators. In: Real-Time Parallel Computing, Springer, pp. 11–35.

[25] He L., Chao Y. & Suzuki K. (2008) A Run-Based Two-Scan Labeling Algorithm. IEEE Transactions on Image Processing 17, pp. 749–756.

[26] Samet H. & Tamminen M. (1988) Efficient component labeling of images of arbitrary dimension represented by linear bintrees. IEEE Transactions on Pattern Analysis and Machine Intelligence 10, pp. 579–586.

[27] Dillencourt M.B., Samet H. & Tamminen M. (1992) A general approach to connected-component labeling for arbitrary image representations. Journal of the ACM (JACM) 39, pp. 253–280.

[28] Image Analysis - Connected Components Labeling. URL: http://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm.

[29] Boswell K.M., Wilson M.P. & Cowan Jr J.H. (2008) A semiautomated approach to estimating fish size, abundance, and behavior from dual-frequency identification sonar (didson) data. North American Journal of Fisheries Management 28, pp. 799–807.

[30] (2019), ARIS File SDK. Contribute to SoundMetrics/aris-file-sdk development by creating an account on GitHub. URL: https://github.com/SoundMetrics/aris-file-sdk, original-date: 2015-11-06T16:53:58Z.

[31] Handegard N.O. & Williams K. (2008) Automated tracking of fish in trawls using the DIDSON (Dual frequency IDentification SONar). ICES Journal of Marine Science 65, pp. 636–644. URL: https://doi.org/10.1093/icesjms/fsn029.

[32] Zivkovic Z. et al. (2004) Improved adaptive gaussian mixture model for background subtraction. In: ICPR (2), Citeseer, pp. 28–31.

[33] Zivkovic Z. & Van Der Heijden F. (2006) Efficient adaptive density estimation per image pixel for the task of background subtraction. Pattern recognition letters 27, pp. 773–780.

[34] KaewTraKulPong P. & Bowden R. (2002) An improved adaptive background mixture model for real-time tracking with shadow detection. In: Video-based surveillance systems, Springer, pp. 135–144.

[35] OpenCV: Background Subtraction. URL: https://docs.opencv.org/3.3.0/db/d5c/tutorial_py_bg_subtraction.html.

Appendices

A. ECHO-SOUNDERS HARDWARE SPECIFICATIONS

The following table holds the physical characteristics of the formed sonograms. The hardware belongs to Sound Metrics Co. [18]. There are multiple other vendors, but in this work, it is mainly focused on pieces of hardware from Sound Metrics. They have two main lines, DIDSON, which is the older, and ARIS which is the newer, where both are mentioned in Table 4. Each family of SONARs has different models, and each model has its own versions. From the table, we can see 8 versions of SONAR, each have their own specifications. The minimum number of beams produced by any of them is 48 beams, while the maximum is 128. When the number of beams increases, the resolution in the horizontal dimension increases as well. Accordingly, Beam width is the angle in degrees between the leftmost and the rightmost edges of any of the beams. While smaller beam widths indicate narrower resultant image, it also indicates higher resolution. Field of View (**FOV**) is the spread of the beams through horizontal and vertical dimensions. It also indicates that there might be a little cross-talk between the beams or blank empty areas instead of overlapping. However, in most cases, the overlapping or blank spaces have negligible widths, hence they are neglected. Range Resolution (**RR**) is the lowest detectable vertical length that the SONAR can identify when the device is used in identification mode on the smallest range. Finally, Maximum Frames per Second (**FPS**) if the number of frames that the SONAR can produce and detect in one second. The higher the number of frames, the more you can detect the full motion and track an object that cuts the SONAR field.

Table 4. Physical sonogram properties from each echo-sounder hardware

Family	Model	Version	Mode Of Operation	Frequency (MHz)	Beam Width (deg.)	Nominal Effective Range (m)	Number of Beams	Beam Spacing (deg.)	Min. Range Start (m)	FOV (deg.)	RR (mm)	Max FPS	Dimensions (cm)	Power (Watts)	Weight (kg)	
															Air	Water
ARIS	VOYAGER	3000	Identification Detection	3.0 1.8	0.3 0.2	5 15	128 64	0.25	0.7	30H*15V	3	15	30*22*17	18	12.2	9.7
	DEFENDER	3000	Identification Detection	3.0 1.8	0.2 0.3	5 15	128 64	0.25	0.7	30H*15V	3	15	43*24*21.5	20	11	N
	DEFENDER	1800	Identification Detection	1.8 1.1	0.3	15 35	96	0.3	-	30H*15V	3	12	43*24*21.5	20	11	N
	EXPLORER	3000	Identification Detection	3.0 1.8	0.2 0.3	5 15	128 64	0.25	0.7	30H*14V	3	15	26*16*14	15	5.12	1.55
	EXPLORER	1800	Identification Detection	1.8 1.1	0.3 0.5	15 35	96 48	0.3	0.7	28H*14V	3	15	31*17*14	15	5.5	1.4
	EXPLORER	1200	Identification Detection	1.2 0.7	0.5 0.8	25 80	48	0.3	0.7	28H*14V	3	15	31*17*14	15	5.5	1.4
DIDSON	DIVER HELD	100	Identification Detection	1.8 1.1	15 35	- -	96	0.3	-	28H	-	15	43*20*17	28	8.1	N
	-	300	Identification Detection Long Range	1.8 1.1 0.7 ∼1.2	0.3 0.4 -	∼35 ∼90 80	96 48 -	0.3 0.6 -	0.42 0.83 -	29H*14V	2.5 10 -	21	31*20.6*17.1	30	7.9	1

B. FILE AND FRAME HEADERS

This appendix holds each of the file and frame headers for ARIS file format. ARIS file format is considered the general layout, where each of the older formats is a subset of its headers.

Table 5. ARIS file header values

Title	Location	Type	Bytes	Description
Version	0	Unsigned Integer	4	Holds the file version number.
Frame Count	4	Unsigned Integer	4	Total number of frames in a given file.
Frame Rate	8	Unsigned Integer	4	Initial recorded frame rate.
Resolution	12	Unsigned Integer	4	0: Low Resolution. Otherwise: High Resolution.
Beams	16	Unsigned Integer	4	Number of beams produced by the hardware.
Sample Rate	20	Float	4	Number of samples captured per second.
Samples per Channel	24	Unsigned Integer	4	Number of samples captured in one beam.
Receiver Gain	28	Unsigned Integer	4	Receiver gains. Ranges from 0 to 40 dB.
Window Start	32	Float	4	Distance between SONAR and first sample.
Window Length	36	Float	4	Distance between first and last samples.
Reverse	40	Unsigned Integer	4	Lens position, either up or down.
Serial Number	44	Unsigned Integer	4	Serial number of the SONAR hardware.
Date	48	String	32	Date of the recording.
Header ID	80	String	256	User input string.
User ID 1	336	Integer	4	User-defined integer.
User ID 2	340	Integer	4	User-defined integer.
User ID 3	344	Integer	4	User-defined integer.
User ID 4	348	Integer	4	User-defined integer.
Start Frame	352	Unsigned Integer	4	First frame number from source file. Normally zero.
End Frame	356	Unsigned Integer	4	Last frame number from source file.
Timelapse	360	Unsigned Integer	4	0: No Timelapse Otherwise: Timelapse recording mode.
Record Interval	364	Unsigned Integer	4	Time between each recorded frame.
Radio Second	368	Unsigned Integer	4	Frames or seconds interval.
Frame Interval	372	Unsigned Integer	4	Time needed to capture one frame.
Flags	376	Unsigned Integer	4	
Auxiliary Flags	380	Unsigned Integer	4	
Sound Velocity	384	Unsigned Integer	4	
3D Flags	388	Unsigned Integer	4	
Software Version	392	Unsigned Integer	4	Version number of the software used for recording.
Water Temperature	396	Unsigned Integer	4	Water temperature code. 0: [5-15]C, 1: [15-25]C, 2: [25-35]C
Salinity	400	Unsigned Integer	4	Salinity code. 0: Fresh, 1: Brackish, 2: Salty
Pulse Length	404	Unsigned Integer	4	Length of the Pulse.
Transmitter Mode	408	Unsigned Integer	4	Transmitter mode in operation.
FPGA	412	Unsigned Integer	4	Reserved.
PSuC	416	Unsigned Integer	4	Reserved.
Thumbnail	420	Unsigned Integer	4	Frame index of frame used for thumbnail image of file.
File Size	424	Unsigned Integer	8	Total file size.
Optional Header Size	432	Unsigned Integer	8	Reserved.
Optional Tail Size	440	Unsigned Integer	8	Reserved
Version Minor	448	Unsigned Integer	4	Hardware version minor release.
Large Lens	452	Unsigned Integer	4	Non-zero if telephoto lens (large lens, hi-res lens, big lens) is present

Table 6. ARIS Frame header

Title	Start	Bytes	Type	Title	Start	Bytes	Type
Frame Index	0	4	uint	Frame Time	4	8	uint
Version	12	4	uint	Status	16	4	uint
Sonar Timestamp	20	8	uint	Day	28	4	uint
Hour	32	4	uint	Minute	36	4	uint
Second	40	4	uint	Hundredth	44	4	uint
Transmit Mode	48	4	uint	Window Start	52	4	float
Window Length	56	4	float	Threshold	60	4	uint
Intensity	64	4	int	Receiver Gain	68	4	uint
CPU Temp.	72	4	uint	PSU Temp.	76	4	uint
Humidity	80	4	uint	Focus	84	4	uint
Battery	88	4	uint	User Defined	92	4	float
User Defined	96	4	float	User Defined	100	4	float
User Defined	104	4	float	User Defined	108	4	float
User Defined	112	4	float	User Defined	116	4	float
User Defined	120	4	float	Velocity	124	4	float
Depth	128	4	float	Altitude	132	4	float
Pitch	136	4	float	Pitch Rate	140	4	float
Roll	144	4	float	Roll Rate	148	4	float
Heading	152	4	float	Heading Rate	156	4	float
Compass Heading	160	4	float	Compass Pitch	164	4	float
Compass Roll	168	4	float	Latitude	172	8	double
Longitude	180	8	double	Sonar Position	188	4	float
Config. Flags	192	4	uint	Beam Tilt	196	4	float
Target Range	200	4	float	Target Bearing	204	4	float

Target Present	208	4	uint	Firmware Revision	212	4	uint
Flags	216	4	uint	Source Frame	220	4	uint
Water Temp.	224	4	float	Timer Period	228	4	uint
Sonar X-Location	232	4	float	Sonar Y-Location	236	4	float
Sonar Z-Location	240	4	float	Sonar Pan	244	4	float
Sonar Tilt	248	4	float	Sonar Roll	252	4	float
Pan PNN	256	4	float	Tilt PNNL	260	4	float
Roll PNNL	264	4	float	Vehicle Time	268	8	double
Time GGK	276	4	float	Date GGK	280	4	uint
Quality GGK	284	4	uint	Number Stats GGK	288	4	uint
DOP GGK	292	4	float	EHT GGK	296	4	float
Heave TSS	300	4	float	Year (GPS)	304	4	uint
Month (GPS)	308	4	uint	Day (GPS)	312	4	uint
Hour (GPS)	316	4	uint	Minutes (GPS)	320	4	uint
Second (GPS)	324	4	uint	Hundredth Second (GPS)	328	4	uint
Sonar Pan Offset	332	4	float	Sonar Tilt Offset	336	4	float
Sonar Roll Offset	340	4	float	Sonar X Offset	344	4	float
Sonar Y Offset	348	4	float	Sonar Z Offset	352	4	float
Trans. Matrix	356	64	float	Sample Rate	420	4	float
Acc. X	424	4	float	Acc. Y	428	4	float
Acc. Z	432	4	float	Ping Mode	436	4	uint

Frequency	440	4	uint	Pulse Width	444	4	uint
Cycle Period	448	4	uint	Sample Period	452	4	uint
Tx	456	4	uint	Frame Rate	460	4	float
Sound Speed	464	4	float	Sample per Beam	468	4	uint
Power Mode	472	4	uint	Sample start delay	476	4	uint
Lens Type	480	4	uint	System Type	484	4	uint
Serial Number	488	4	uint	Reserved	496	8	uint
Error Flags	500	4	uint	Missed Packets	504	4	uint
App. Version	508	4	uint	Reserved	512	4	uint
Order	516	4	uint	Salinity	520	4	uint
Pressure	524	4	float	Battery	528	4	uint
Main Voltage	532	4	float	Switch Voltage	536	4	uint
Autofocus	540	4	uint	Voltage Change	544	4	uint
Focus Timeout	548	4	uint	Focus Over Current	552	4	uint
Focus Not Found	556	4	uint	Focus stalled	560	4	uint
FPGA Timeout	564	4	uint	FPGA Busy	568	4	uint
FPGA Stuck	572	4	uint	CPU Temp.	576	4	uint
PSU Temp.	580	4	uint	Water Temp.	584	4	uint
Humidity	588	4	uint	Pressure Fault	592	4	uint
Voltage Read	596	4	uint	Voltage Write	600	4	uint
Focus Position	604	4	uint	Target Pan	608	4	float
Target Tilt	612	4	float	Target Roll	616	4	float

Pan Motor Error	620	4	uint	Tilt Motor Error	624	4	uint
Roll Motor Error	628	4	uint	Pan Absolute Position	632	4	float
Tilt absolute position	636	4	float	Roll absolute Position	640	4	float
Pan Accel. X	644	4	float	Pan Accel. Y	648	4	float
Pan Accel. Z	652	4	float	Tilt Accel. X	656	4	float
Tilt Accel. Y	660	4	float	Tilt Accel. Z	664	4	float
Roll Accel. X	668	4	float	Roll Accel. Y	672	4	float
Roll Accel. Z	676	4	float	Applied Settings	680	4	uint
Constrain Settings	684	4	uint	Invalid Settings	688	4	uint
Interpacket Delay	692	4	uint	Interpacket Delay Period	696	4	uint
Uptime	700	4	uint	Version (Major)	704	2	uint
Version (Minor)	706	2	uint	Go Time	708	8	uint
Pan Velocity	716	4	float	Tilt Velocity	720	4	float
Roll Velocity	724	4	float	GPS Time	728	4	uint
System Variant	732	4	uint	reserved	736	-	

C. OUTPUT TEMPLATES

This appendix holds a template of the output in `JSON` format, and the legacy text template produced by Sound Metrics software after manual counting.

```
{
    "key": {
        "ID": int(),
        "locations": list(),
        "frames": list(),
        "area": list(),
        "bounds": list(list())
    },

    "processingDate": str(),
    "analysisParameters": {
        "kernelDimensions": list(),
        "kernelShape": list(list()),
        "startFrame": int(),
        "blurDimensions": list(),
        "threshold": int(),
        "minAppearance": int(),
        "maxAppearance": int(),
        "searchRadius": int()
    }
}
```

```
Total Fish          =       [integer]
Upstream            =       [integer]
Downstream          =       [integer]

Total Frames      = [integer]
Expected Frames   = [integer]
Total Time        = [time]
Expected Time     = [time]

Upstream Motion   = [string]

Count  File Name:  [string]
Editor ID          = [string]
Intensity          = [float]
Threshold          = [float]
Window Start       = [float]
Window End         = [float]
Water Temperature  = [integer]
```

```
File : [integer]
Total : [integer]
Frame# : [integer]
Dir : [string]
R (m) : [float]
Theta : [float]
L(cm) : [float]
dR(cm) : [float]
L/dR : [float]
Aspect : [float]
Time : [time]
Date : [date]
Latitude : [string]
Longitude : [string]
Pan : [float]
Tilt : [float]
Roll : [float]
Species : [string]
Motion : [string]
Q : [integer]
N : [integer]
Comment : [string]
```